NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

VISUAL PLANNING AID FOR MOVEMENT OF GROUND FORCES IN OPERATIONS OTHER THAN WAR

by

Norbert Schrepf

March 1999

Thesis Advisor:	Arnold Buss
Second Reader:	Gordon H. Bradley

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. **AGENCY USE ONLY** ***(Leave blank)***	2. **REPORT DATE** March 1999	3. **REPORT TYPE AND DATES COVERED** Master's Thesis

4. **TITLE AND SUBTITLE** VISUAL PLANNING AID FOR MOVEMENT OF GROUND FORCES IN OPERATIONS OTHER THAN WAR	5. **FUNDING NUMBERS**
6. **AUTHOR(S)** Norbert Schrepf	
7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000	8. **PERFORMING ORGANIZATION REPORT NUMBER**
9. **SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Air Force Office of Scientific Research, 110 Duncan Avenue, Suite 100, Bolling AFB, DC-0001	10. **SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. **SUPPLEMENTARY NOTES**
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. **DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited.	12b. **DISTRIBUTION CODE**

13. **ABSTRACT**

The fall of the Berlin Wall in 1989 marked the change of the political and military situation worldwide. Peace keeping missions became more likely than major regional conflicts. However, the conventional combat simulations, which were developed for the combat between heavily armored forces could not handle these new situations. In these new missions the movement of ground forces becomes a major task for any commander. This thesis develops a software architecture of loosely coupled software components. These components are combined to simulate the movement of convoys. The simulation is implemented as an event step model. For visualization of the ongoing simulation a different component displays the convoy locations on a geographical display. The combination of both modules allows the analyst to validate a given movement plan and to identify possible weak points and threats.

14. **SUBJECT TERMS** Operations other than War, Peace Keeping, Movement Planning, Convoy, Ground Forces, Simulation, Software Components	15. **NUMBER OF PAGES** 100
	16. **PRICE CODE**

17. **SECURITY CLASSIFICATION OF REPORT** Unclassified	18. **SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	19. **SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	20. **LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

VISUAL PLANNING AID FOR MOVEMENT OF GROUND FORCES IN OPERATIONS OTHER THAN WAR

Norbert Schrepf
Captain, German Army
M.S.(C.S.), University of Federal Armed Forces, Munich, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1999

Author: [illegible]
Norbert Schrepf

Approved by: [illegible]
Arnold Buss, Thesis Advisor

[illegible]
Gordon H. Bradley, Second Reader

[illegible]
Richard E. Rosenthal, Chairman
Department of Operations Research

ABSTRACT

The fall of the Berlin Wall in 1989 marked the change of the political and military situation worldwide. Peace keeping missions became more likely than major regional conflicts. However, the conventional combat simulations, which were developed for the combat between heavily armored forces could not handle these new situations. In these new missions the movement of ground forces becomes a major task for any commander. This thesis develops a software architecture of loosely coupled software components. These components are combined to simulate the movement of convoys. The simulation is implemented as an event step model. For visualization of the ongoing simulation a different component displays the convoy locations on a geographical display. The combination of both modules allows the analyst to validate a given movement plan and to identify possible weak points and threats.

THESIS DISCLAIMER

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

When the Berlin wall came down in 1989 and the cold war between East and West ended, the face of the military world changed dramatically. Up to this time NATO's strategic thinking on the battlefield in Europe was mainly influenced by high-intensity combat and heavily armored divisions. With this new situation in Europe and around the world it now had to change to low-intensity combat and peace keeping operations. A major example of this is the civil war on the territory of the former Yugoslav Republic. While the civil war in Croatia could be settled fairly quickly, the civil war in Bosnia/Herzegovina went on for years.

Since its establishment in 1992, the HQ ACE Rapid Reaction Corps (ARRC) was preparing plans for a peacekeeping operation under NATO command on the territory of the former Yugoslavia. While the aim of such an operation constantly changed due to higher political decisions, it became rapidly clear that one of the major problems in this operation would be the movement of ground forces at the beginning and end of the NATO engagement. For this purpose the HQ ARRC, Operational Analysis Branch, developed a set of software tools in conjunction with the Cranfield University in England to simulate the movement of ground forces. However, the NATO operation in Bosnia Herzegovina (BH) began before the movement simulation could be finished and it was finally discarded.

This thesis develops a loosely coupled components architecture for software model development, which allows individual independent modules to interact with each other. This enables analysts to program small software components individually which allow them

to focus on one specific task at a time. The final combination of these modules provides a powerful tool combining all the advantages of the individual modules. As an application, a movement simulation for ground forces was developed. For modeling purposes the moving units are represented as convoys combining a number of vehicles together in one entity of the simulation. As in the military planning process the route network only represents a pre-selected number of roads which the convoys may use so they can reach their target. The simulation was developed as an event step model using the SimKit package, developed at the NPS, for implementation. Java™ was chosen as the programming language to achieve the maximum portability for the software between various computer platforms. For visualizing the results of the simulation a pure geographical display tool was implemented which allows various levels of resolution.

The combination of both modules within the loosely coupled software architecture provides a powerful tool for visualizing the effects and results of a movement plan. The simulation can be stopped at any time and this enables the user to examine the locations of the moving convoys in relationship to each other. As a result the complete movement plan can be evaluated and weak points or high risk operations can be visualized immediately.

ACKNOWLEDGEMENT

The author would like to acknowledge the support of the Micro Applications Group at Cranfield University, UK, in programming assistance and for technical advice on Project THISTLE. A very special 'Thank You' to my 'mentor' James Ford, PhD, who advised me in spite of distance and time differences and readily supported all my activities. I also would like to thank HQ ARRC, Operational Analysis Branch for supplying the map data and allowing me to use the map conversion software OAK.

Another special 'Thank You' belongs to my thesis advisor, Prof. A. Buss who was always willing to listen to my ideas and was prepared to discuss these with me.

I. INTRODUCTION

A. THE POST COLD WAR ERA

In November 1989, when the Berlin Wall came down and the Iron Curtain in Europe disappeared, the military world changed quite dramatically. But the earth didn't become a much more peaceful place. New conflicts appeared on the political stage that were substantially different from the scenarios which were assumed before 1989. Up to then, most planning was concentrated on major regional conflicts between two heavily armored forces using a wide spectrum of weapon systems with very few restrictions on how to use them. Now likely scenarios are local conflicts and civil wars, such as the conflicts in Cambodia, Somalia, Rwanda and the former Yugoslavia. The type of military action has changed from heavy fighting, with the goal to destroy enemy forces, to peace reestablishment, peacekeeping, and saving of lives. These were tasks for which the involved armies had little or no training. So far the military operations research (OR) community has not concentrated on these kinds of missions and therefore has only a very limited set of tools to support the type of operations required for carrying them out. Examples of these new types of operations under military control include:

- movement control of refugees and humanitarian aid convoys conflicting with military operations,
- stabilization of the political and military situation within a country or a region,
- preparation of elections,
- reinstallation of self governmental control.

This thesis develops a set of tools which will support the planning of humanitarian operations involving movement of refugees, thus helping to fill the gap in OR support of these missions. The usefulness of these tools will be shown in a simulation of a scenario, which is similar to the one, which was exercised by higher NATO commands in 1995.

B. SCENARIO

In 1992 the ACE Rapid Reaction Corps (ARRC) was established as a reaction force which could be used by the Supreme Allied Commander Europe (SACEUR) in response to crises within his area of responsibility. The multinational headquarters (HQ) of this reaction force is capable of commanding up to four divisions in an operation. These divisions are assigned to HQ ARRC by NATO nations for specific missions.

Immediately after being established, the HQ ARRC started considering possible options for a peacekeeping operation on the territory of the former Yugoslavian Republic. It thereby focused especially on a peacekeeping mission under NATO command within the boundaries of Bosnia-Herzegovina.

In February 1995 a possible withdrawal operation of UN forces under NATO protection from Bosnia-Herzegovina was exercised. HQ ARRC was the supreme ground HQ in this exercise. The higher command for HQ ARRC during this exercise was HQ Allied Forces South (AFSOUTH). HQ AFSOUTH later became HQ Implementation Force (IFOR) during the peacekeeping operation in Bosnia/Herzegovina in 1995/96. For this exercise, the HQ was deployed under realistic conditions with its operational equipment. For the training exercises, HQ ARRC had three divisions under command:

- 1st US Armored Division,
- 3rd UK Mechanized Division and
- 6th French mechanized division.

The HQs of these divisions were also involved in this exercise.

The exercise DETERMINED EFFORT 95, took place at the Warrior Preparation Center, Ramstein, Germany. Within this exercise it became clear that the old "general region conflict" combat simulations could only be used to a very limited extent for planning and evaluation of this kind of operation. The reason for this was that this new type of operation consisted primarily of moving units and refugees, negotiations with the warring factions, and demonstration of power. Very little fighting was expected and, if there was any, it would be guerrilla actions against UN or NATO forces and convoys resulting in only a few shots of sniper fire or mortar rounds.

A combat simulation for major regional conflict was used as an exercise driver. Shortly after the start of the exercise it became apparent that this program was not able to handle the scenario. Each time opposing forces came into fighting distance, the simulation computed a combat situation which resulted in the extinction of one of the fighting units. For these combat situations the simulation computed heavy losses on UN convoys, which consisted mainly of commercial trucks and no fighting equipment. This was an unrealistic and unexpected result. The whole system lost its credibility during this exercise and had to be reprogrammed.

C. AVAILABLE PLANNING TOOLS

In February 1995 during the preparation for a NATO operation on the territory of the former Yugoslavia, the HQ ARRC exercised the withdrawal of UN forces under NATO protection at the Warrior Preparation Center in Ramstein, Germany.

Attached to HQ ARRC was a Operational Analysis Branch (OAB) which consisted of three civilian scientists trained in OR and two officers who had the role of military advisors. During the previous four years of monitoring the civil war by UN forces, there were considerable quantities of data available on the operational route network. Staff members of HQ ARRC went on reconnaissance tours to UN forces in the area of interest to prepare for possible operations. While these data were not nearly complete and changed daily, they provided the basis for planning during this exercise. There also was a high degree of uncertainty about what would actually happen on the ground and how the weather might influence the status of the available roads.

Very early during the planning it became clear that one of the major areas of uncertainty was the movement of ground forces. While very little fighting was expected against NATO and UN forces, there was still the fear that isolated UN soldiers might get taken hostage by the warring factions and used as a shield for self protection inside Bosnia/Herzegovina (BH). It was therefore critical to quickly move a strong force very fast into BH and to recover all isolated UN units in as fast and non-violent manner possible. Because of the uncertainty of the road conditions and the restrictions on movement speed within BH, air recovery of UN units and abandonment of all heavy equipment was considered.

The planning cycle of a corps HQ lasts about 72 hours. This means that the HQ plans its operations ahead of time and notifies its subordinate units about its planned actions. The task of the OAB within HQ ARRC was to support all staff branches in operational planning. However, the majority of work during this exercise was done for the HQ ARRC movement branch, which estimated the expected movement of its own forces. At this time there were only very limited tools available to OAB. The major planning tool was an EXCEL spreadsheet, which was constructed during this exercise. A simple movement simulation called BIEST was also available for planning purposes, but this was difficult to use and could only be operated on a SUN/SPARC workstation. However, even which its limited capabilities, this tool proved valuable, and OAB was tasked to develop a complete simulation model for movement of ground forces after the end of the exercise. Cranfield University was tasked to develop the movement simulation BIEST further and to integrate with other planning and staff tools. Due to priority changes in the overall contingency planning of HQ ARRC in respect of a BH operation, only the first stage of this model was developed; it was never finished.

Since "operations other than war" (OOTW) are more likely nowadays, there is a desperate need for models to support and evaluate planning. The movement planning in any deployment, redeployment, or withdrawal phase is absolutely critical. With an increasing number of entities, however, it is difficult to forecast the influence of large refugee movements and vulnerable route networks.

The Operations Research Curriculum at the Naval Postgraduate School (NPS) uses a Java based simulation kit, called SimKit, in the System Simulation class. SimKit allows the easy development of event step related models (Buss 1996). This tool provides a good foundation for a movement simulation which could support a planning procedure as described above. However, the current SimKit classes lack the capacity to visualize the computed events on a geographical display. A geographical display tool would be most useful for the user and analyst to visualize the ongoing simulation.

D. OUTLINE OF THE THESIS

This thesis develops a graphical movement simulation, which allows the analyst to visualize the effects of the movement of ground forces on a geographical display. The system which was used by OAB during the exercise was a monolithic software product in which display and simulation where integrated within the same package. While the system architecture was developed by OAB, the implementation of the software was done at Cranfield University. One of the major problems it that it is difficult to include new software tools or to change the existing implementation. These changes could only be done at the university where the complete source code was located. The time for bug fixes and new features therefore was sometimes more than a month, which is not acceptable when rapid planning is required. To address this problem the software architecture developed in this thesis has a very loose structure of independent, small components. Each of these is capable of running independently from the other components and can produce results on its own. A movement simulation, which just produces as output a long list of locations of convoys, cannot be called a user-friendly and helpful tool. Without a visualization of the locations of the individual convoys involved in the simulation the results cannot easily be verified or used. If this list of locations is displayed on a geographical map display, then the results of this simulation can be easily seen and analyzed by planners and analysts. A loose coupling of components allows the movement simulation to be written totally independent from the map display. Also the user has the benefits of a wider range of display options.

The software model and architecture for this thesis is therefore organized in the following way:

1. pure movement simulation which uses SimKit,
2. geographical map display tool, which will be capable of showing geographical related data on a map display.

To demonstrate the system, a simulation is used to evaluate a movement plan of one division in a scenario similar to that used in the HQ ARRC exercise for a possible withdrawal of UN forces under NATO protection.

E. RELATED RESEARCH

1. Cranfield University

From 1992 to 1995 the Micro Processor Application Group of Cranfield University developed in conjunction with the HQ ARRC, Operational Analysis Branch a set of software tools for command and control purposes in preparation for the IFOR operation in Bosnia/Herzegovina. The project was called THISTLE and its purpose was to serve as a technical demonstration of a future command and control (C2) system in a modern HQ. The system was used successfully in 1995/96 during the NATO operation in Bosnia-Herzegovina and the prototype software was adopted as the C2 software until the future introduction of a 'official' C2 software package.

The basic idea of THISTLE is to group a set of independent software modules and tools around a map display tool called FLORA. The system is capable of showing geographical related data from various sources on one common map display. One aim was that all the involved software models had the capability to run in a stand-alone mode. They also have the capability to exchange information among the models and to display tactical information on a common map display tool. The system was rapidly developed within six months by using existing software tools. Due to the shortage of time not all systems could be integrated and so the engineering software which was manipulating the route network was totally independent from the rest of the THISTLE environment. However, most of the other tools were capable showing geographical referenced data on FLORA. Another major problem was that the software was developed for the

SUN/Solaris operation system and each development step had to be converted to a Windows 3.11 version; this proved to be very time and manpower consuming. A major problem was that the two systems were not 100% compatible after the conversion. For example, the map database was different in each system and so two different databases had to be prepared, installed, and maintained. Maintainance of the entire software package was not an easy task.

One of the models developed for the Operational Analysis Branch was the movement model IVY, which simulates the movement of convoys on a predefined route network. With IVY it was possible to develop a complete movement plan in advance and then run the situation and watch the results on a map background. However, the model was time step driven and did not focus on events as they occur. This meant that some of the results were crude and different length of the time steps could produce different results for the same simulated scenario. One of the major problems was the update rate. For each convoy the current position had to be calculated at every time step. Even if no convoy was moving, nearly the same calculation time was needed to update the display. For scenarios over a long time period this implied a long calculation time, because the length of the time intervals were given at the beginning of the simulation. Therefore, there was a tradeoff between the run-time of the model and accuracy of the actual convoy positions, which represent the situation on the ground.

The author of this thesis was the project officer for the development of project THISTLE in OAB and Cranfield University. Because the IVY movement model was never finished, there were also no reports produced on the actual project development.

2. Optimization Approach

The article "Optimizing Disaster Relief: Real-Time Operational and Tactical Decision Support" (Brown, Vassiliou, 1993), introduced a real-time decision support system using optimization methods, named ARES (for the Greek god of war). The idea of ARES is that after some kind of disaster there are a number of tasks which have to be done. The decision maker controls some units and resources which can be assigned to fulfill these tasks. ARES supports the decision about which unit to assign to which task and which resources should be allocated to individual units. The system is implemented in such a way that it allows the user to quickly build a scenario for which some decisions have to be made. ARES optimizes the available resources and presents the results so the decision maker can understand and accept all or any part of it. While ARES is only a decision support system, the user is required to apply her/his own judgment to the results before making the final decision.

ARES is based on three linear programs. Two of these linear programs deal with the operational assignment of units to tasks. The idea is to assign only one unit to each task and to minimize the travel distance for each unit. The third linear program optimizes the efficiency of resources allocated to each task.

The strength of ARES is its capability to optimize resources to complete a number of tasks. However, it is only using a very crude model of representing the available road network on which units can travel. Further it is assuming that there is no problem with several units trying to travel along the same road and there is no blocking between units at road sections. This seems reasonable for this kind of system; the time frame it is dealing with is several weeks to complete a task and therefore some delay of moving units will not have a major impact on the overall result. However, in a scenario such as that described in chapter I.B where the speed of movement is critical for the success of the overall operation and where the road conditions and movement conflicts of units are important, ARES cannot inform the decision maker on how to move the available units into place. The logical extension of ARES therefore is to feed its results in a movement simulation such as developed in this thesis. The movement of the ground forces can be visualized and the decision maker can see a picture of the impact on unit movements by allocating tasks to units.

II. FOUNDATIONS

A. MOVEMENT OF GROUND FORCES

Movement of forces is one of the military tasks common to most scenarios involving ground forces. In any operation a commander will always try to concentrate his forces to maximize their effectiveness. Also the commander will always try to prevent its opponent from moving forces into a superior position. During the Gulf War in operation Dessert Storm the allied commander, General Schwarzkopf, moved most of his forces to the left flank of his attacking forces and gained superiority there. This movement was done in a very short time frame hidden from the reconnaissance of the enemy.

In operations other than war (OOTW) the movement of ground forces is still one of the major concerns of any ground commander. At the beginning of an operation it is essential to reach all the key position within the area of interest as fast as possible. However, this movement is normally visible to all the warring factions. It also might conflict with civilian traffic and the movement of other forces located in the area of operation. Because of the nature of the mission, the peacekeeping forces do not have total control over the route network in general. Quite often there are many restrictions such as civilian traffic regulations. This was a major problem in BH before the IFOR operation, since UN forces had to use a route network, which was controlled by the warring factions. Some routes could only be used during a certain time of the day. On other routes only a certain number of vehicles were allowed to use a given road. Depending on the intensity of fighting between the warring factions there might also be a considerable number of refugees trying to escape from the combat areas and therefore moving along the available routes. Refugees quite often use various kind of vehicles which cannot be ignored as a movement obstacle for one's own forces. These refugee convoys typically move much more slowly than the desired speed of military forces, causing traffic jams and, in the worst case scenario, a objective might not be reached in time and a whole plan might fail.

B. ROADS

Ground forces use the existing roads within the area of operation for movement between two points. Although off-road movement is possible for a large variety of military vehicles, it is only done if absolutely necessary, due to the extra cost in terms of time, speed and fuel consumption. Military trucks which have off-road capabilities typically use it to bypass obstacles on a road. Even combat vehicles like tanks and infantry fighting vehicles (IFV) drive cross-country only when it is more important to use cover rather than good movement conditions.

Because this thesis addresses the movement of ground forces in OOTW, off-road movements are ignored, with the only exception being bypassing of obstacles. This situation is modeled by limiting the maximum travel speed on a road for the distance of the obstacle, with the effect that a convoy will be slowed down.

C. ROUTE NETWORK

For planning purpose military HQs organize the movement of ground forces in the area of operation on a route network. An individual route consists of a series of connected roads. Quite often individual routes are assigned names for convenient reference. The collection of all pre-selected routes form the available route network. For the UN operation in BH there was a observed route network which was used by convoys of UN forces. Although this route network was partially maintained by UN troops, parts of it were under control of the warring factions, which only maintained the road sections essential for their own movement operations. Road sections crossing the line of confrontation were in especially bad condition and sometimes could not be used at all. The selection of routes however changed only very rarely over the time of the IFOR operation. Figure 1 shows a small area of this route network which is taken from the BH Command Route Information map, dated December 5th, 1994.

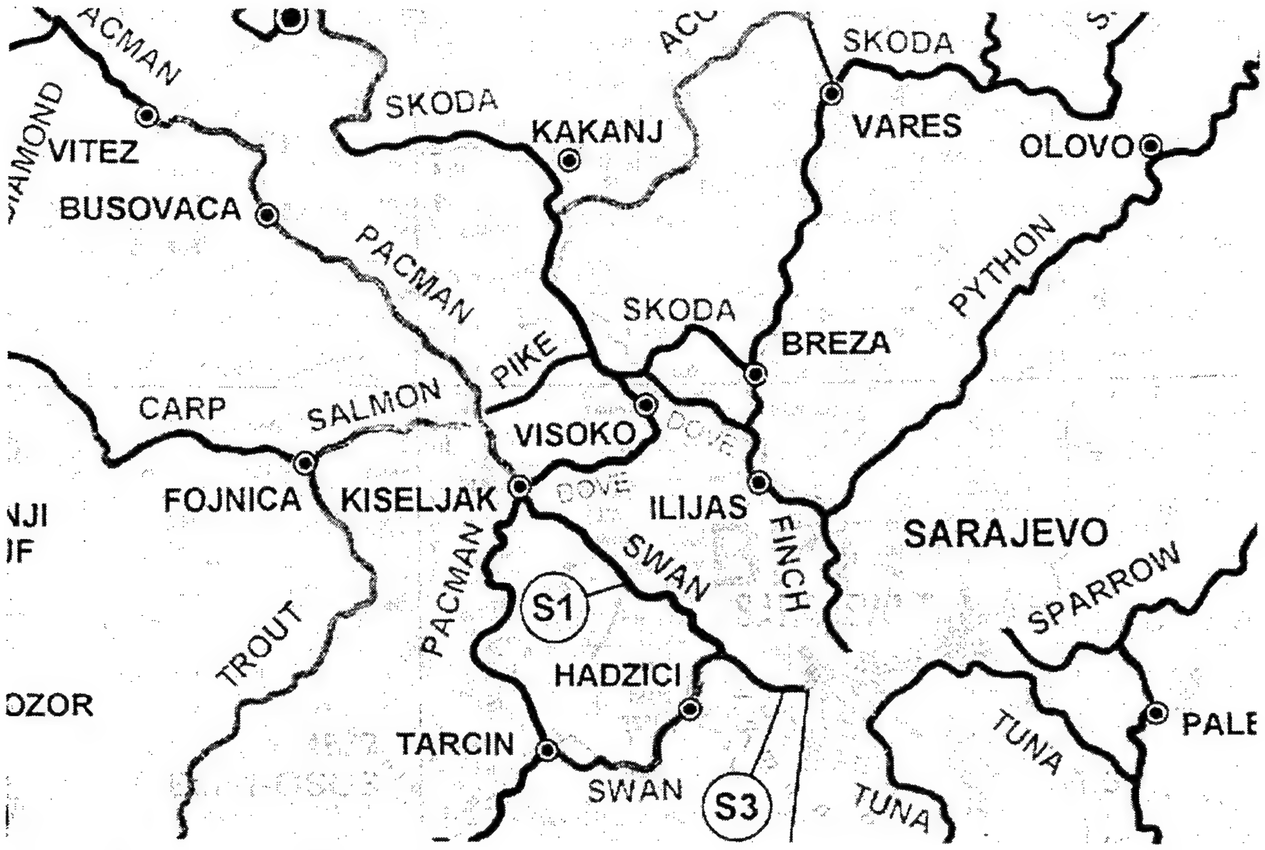


Figure 1: Example of the BH Route Network.

Because of the very mountainous terrain in BH, most of the route network was not everywhere as dense as in the area around Sarajevo. Off-road movement was nearly impossible in most parts of BH, and some areas could only be reached on one route. This was especially true in the Serb controlled areas of BH. If a route was interrupted by any circumstances it was quite often necessary to deliver supplies by airlift. Checkpoints of the warring factions along the roads also influenced the movement of ground forces quite a lot. However, the Dayton Peace agreement forced the freedom of movement in the whole of BH and therefore these checkpoints disappeared with the beginning of the IFOR operation.

D. CONVOYS

For self-protection and organizational reasons military vehicles normally travel in convoys that contain from three up to one hundred and possibly even more vehicles. The operation of military convoys is very similar in all armies around the world. At the beginning the vehicles are lined up in an assembly area from where they enter the road at the start of the travel. The convoy leader is normally driving inside the first vehicle from which he tries to control the movement of the whole convoy. The average length of a vehicle within a convoy is approximately 10 meters. The leader determines the distances between vehicles and orders the normal and maximum speed for the travel. These parameters depend very much on the road condition and on the existing threat to the convoy by air and ground forces. While in OOTW the air threat is certainly close to zero, there is a constant threat from snipers and nuisance fire by mortars and guided missiles. Some realistic parameters for a convoy traveling on country roads and tracks are:

- distance between vehicles: 50 to 100 meters,
- normal convoy speed: 15 to 40 km / hour,
- max. speed for closing up 25 to 50 km / hour.

The German Army uses a distance of 100 meters between vehicles for its convoy moment in training and exercise outside built-up areas and 50 meter inside towns and villages. Because of the permanent threat of sniper and mortar attacks, however, it makes sense not to reduce the distance between vehicles in built-up areas. This reduces the risk that one single mortar shell can damage more than one vehicle.

Figure 2 shows the theoretical structure of a convoy and indicates parameters that determine the convoy's length.

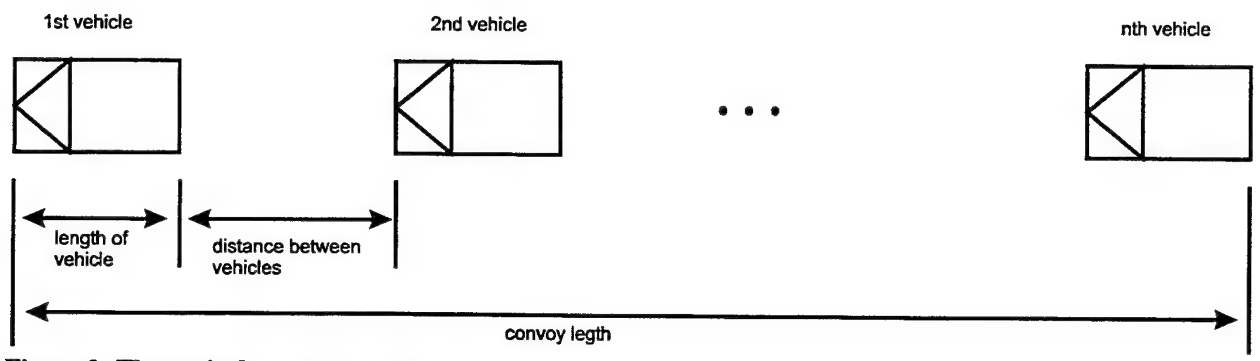


Figure 2: Theoretical convoy structure.

While the length of a convoy consisting of only a few vehicles is not critical for planning reasons, a convoy of 50 vehicles occupies approximately 5500 meters on a road. The time it takes a convoy to bypass a given point on a road can easily calculated with the formula in equation (1).

n	number of vehicles in a convoy,
d	distance between vehicles
l	average length of a vehicle,
v	convoy speed,
t	time to bypass a certain point on a road.

$$t = \frac{n \cdot (d + l)}{v} \qquad (1)$$

Assuming a convoy speed of 25 km/hr under the previous conditions, a given road point (e.g. crossing) would be blocked for more than 13 minutes. However, the equation (1) is only valid if the convoy is not changing speed during the time it is passing the observed point on the road. If the convoy changes its speed during the passing event, this time becomes either longer or shorter depending whether the convoy increases or reduces its speed.

It is also important to realize that the first vehicle does not normally have contact with the vehicles behind other than through the rear view mirror. While every vehicle will try to follow the vehicle ahead of it at the ordered distance, this might not always be possible, especially if road conditions restrict the speed which a vehicle is allowed to drive. Therefore sometimes a vehicle is driving at the ordered normal convoy speed while the following vehicle is still driving through a 'slow-go' area, in which the maximum driving speed is lower due to road conditions. This results in a greater distance between the vehicles impacting the total length of the convoy. Calculations about the length of a real convoy are especially complicated because they also depend heavily on the reaction and behavior of the human drivers. The distance between vehicles will always be very flexible and only roughly approximate the ordered distance. This distance also should include a buffer so that small speed changes of preceding vehicles do not have a major effect on the following vehicles.

For counteracting length changes a sensible leader will also order the speed of the convoy head to be a certain percentage below the maximum possible speed on that road. The vehicles inside the convoy body can keep the ordered distance using a follow-up speed that is higher than the speed of the first vehicle of the convoy. This way the change in length for a convoy can be kept to a minimum. The better the drivers are trained for convoy travel, the less impact there will be on the length of the convoy due to speed changes of the first vehicle. Because these changes in length now only occur for short periods of time under these conditions this effect is ignored in the simulation and the convoy length is assumed to be constant.

III. MODEL FEATURES

A. MODELING APPROACH

In the past software models were designed to serve a specific purpose. Most of the functionality was build into the system and only very little functionality was imported from outside. Quite often these systems were built by using one of the common programming languages such as BASIC, C or FORTRAN together with a database package which are platform dependent. This approach resulted in different implementations of interfaces all of which had similar or even identical tasks. Therefore a considerable amount of time must be spent each time just to learn how to handle the interface before the tool can be used. While most simulation models have many things in common, such as the geographical display, there has been only a very limited effort to build systems out of exchangeable components which can be replaced easily without effecting the functionality of other modules.

The current way of implementing models results in long development times. Sometimes similar components of a simulation or model have to be rewritten again and again just to perform the same task. The best example for this is the geographical display with its functionality of zooming and display of tactical data in form of standardized tactical symbols. Another major problem area is the duplication of code in order to perform similar tasks in different tools. It is very difficult to maintain code which has been duplicated over a variety of software products. If this code then also gets modified in some of the tools, it cannot be easily replaced by just overwriting the existing code as updates get available. From this point onwards this code section has to be handled as independent and therefore has to be maintained as such. The maintenance of this functionality within several tools now becomes a major task which has to be dealt with for each individual tool.

This modeling approach has resulted in huge, independent software packages for which the exchange of data with outside models is very difficult task. A common display of the results of more than one model at the same time is for most systems nearly impossible.

To overcome the shortfalls of the previous described software architecture, there is the concept of a modularized set of software components which all work independently, but can also be connected in some way so they can perform more complex tasks. This architecture is described in a paper by Bradley and Buss about an architecture for dynamic planning systems using loosely coupled components (Bradley/Buss 1997 and Arntzen 1998).

The idea is to develop software modules which are only coupled very loosely, and are still independent enough that they can perform tasks on their own without the need for any other components. Each component should only concentrate on solving a very specific task. The benefit of bringing two of more of these components together would be that in conjunction they can solve more complex problems and benefit from each other's functionality. A loosely coupled component architecture of software modules would then allow a software developer or analyst to concentrate just on the specific task. A user would benefit from more standardized interfaces which can be applied to many tools and models. In the specific example about 'movement of ground forces' a user would need a geographical display tool and a movement simulation. Both should be individual packages, but also should be able to communicate with other software via a standardized interface. Like a set of LEGO blocks the user can assemble a system "on the fly" which serves her/his specific needs.

A graphical overview of the implemented software architecture is shown in Figure 3. The components represent individual software packages which are developed to solve specific tasks with little or no extra functionality. Examples for these components include geographical display tool, movement simulation, tactical overlay drawing component, and an overlay slide show component. The message center is the heart of the software architecture, providing the required functionality so that the individual components can communicate with each other. Any component can send messages to the message center, but if it wants also to receive messages it has to register there first.

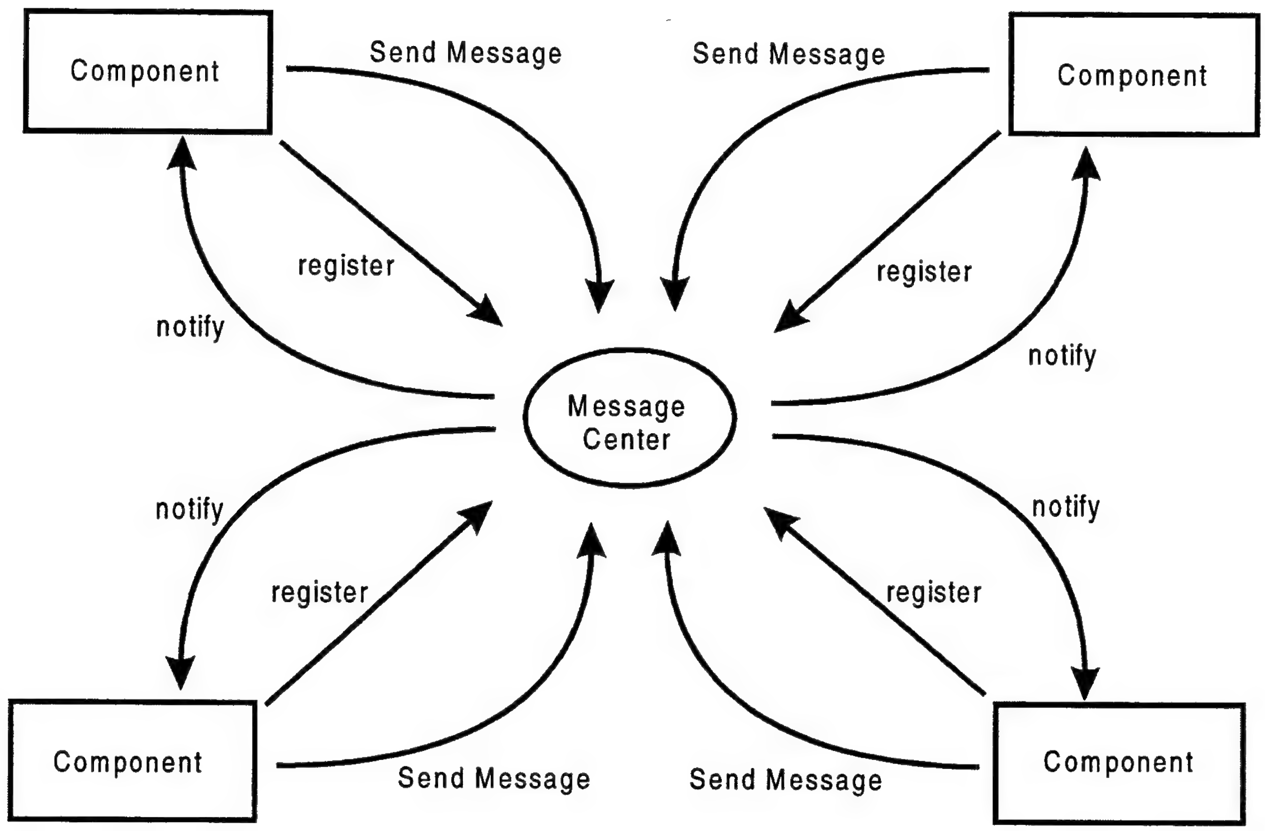


Figure 3: Software architecture.

The flow of any message follows an easy pattern. A component creates a message and sends this message to the message center with no specified receiver. The message center takes the message and multicasts it to all registered listeners. It is then the task of the notified component to process the message any further. The component will determine from the content of the message if it is in a form it can process any further and in this case handle it. Otherwise the message will just be ignored. This way the sending component does not need to have any knowledge about who is listening to the message center. It is just producing its own results and offers them in a message to other components for further handling. This concept offers maximum flexibility for developing new software components independently from the existing system. The only connection between these components are message objects, which are handed over by the message center. This way components do not need any knowledge about each other and can be developed and compiled totally independent.

Under some circumstances a component might want to send a message direct to a specific component. This could be the case if it gets a message for which it needs more information or if it wants to send a processed message back to the originating component. This can be done by specifying the receiving component, in which case the message center will not broadcast the message but instead just deliver it to the specified component. Even in this case the component does not have the need for specific knowledge about the component which was originating the message. All it needs to know is that it received a message and it wants to send another message back to the sending component. The message center will take care about routing the messages.

This concept is very powerful and allows the development of new components and the replacement of existing components at any time without affecting the rest of the system. A user can now just pick her/his favorite tools and operate them together to perform a specific task.

For the simulation of the movement of ground forces in OOTW two major components have been developed. A pure non-visual movement simulation called IVY simulates the movement of convoys in an event model. The tactical data which represents the symbols and locations of the convoys is transferred via the message center. The

Common Data Transfer Format (CDTF) as used in HQ ARRC and NATO C3 Agency, The Hague, NL is used as a means to represent and transfer this data (Schrepf, 1996). This allows a flexible transfer of tactical information in the NATO standardized form of APP-6 (SHAPE 1984). After fixed time intervals the locations of all simulated convoys are send by IVY in the CDTF format to a map display tool called FLORA. FLORA then represents this information on a geographical background. This way simulation and graphical user interface (GUI) are separated from each other.

B. ROUTE NETWORK

The basis for the movement simulation is the available route network. This determines the available movement paths for convoys participating in the simulation. The route network consists of roads and junctions. The following section explains how this is modeled in IVY.

1. Roads

Roads are modeled as the connection between two junctions. This restricts a road to be straight between these two points, which is very seldom true in reality. Therefore a road is broken up into smaller sections. Road sections along a winding road are connected with two way junctions under this circumstances. This enables modeling the path of a road much more accurately. If a road is divided into sufficiently small sections a very detailed model of the road can be achieved. Figure 4 shows a example road with three road sections connected by junctions.

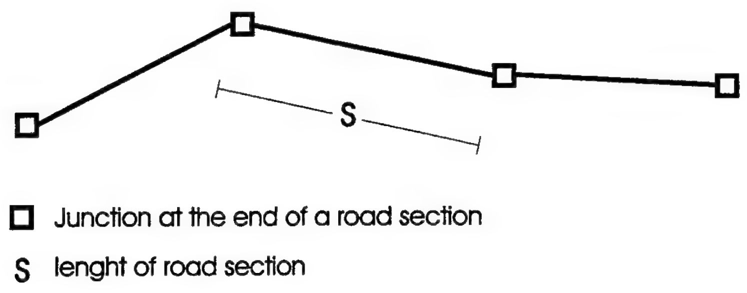


Figure 4: Road model.

The length of a road is given by the sum of the lengths of the individual road sections. For calculating the length of a road section the distance between the two endpoints in a two dimensional space is calculated. Note that this ignores the fact that there might be a height difference between the two endpoints as well as the curvature of the surface of the earth. For the convoy planning problem dealt with, this is an acceptable simplification and does not introduce considerable error. Furthermore, adding elevation to the road segments is a straight forward matter.

The combination of roads and junctions forms the available road network. For modeling reasons it is assumed that a convoy can only enter a road at a junction. This is not really a restriction, because any road section can be split up into two sections connected with a two way junction and so allow a convoy to start at any point of the road network.

2. Convoy start and end location

A convoy occupies a certain part of the road network. The determining points of a convoy are the location of the first vehicle and the location of the last vehicle, as shown in Figure 5. The vehicles in between do not have to be modeled in all detail under the assumption that a convoy will not be split up while it is traveling and other convoys are not allowed to infiltrate a moving convoy. These are realistic assumptions and do not restrict the way the military is operating the movement of convoys. From the location of the first and last vehicle of a convoy the exact part of the occupied road section can be determined. In order to describe the state of a convoy completely the speed of the first and last vehicle have to be noted. For simplification the velocity of the last vehicle is set equal to the velocity of the first vehicle. This assumption was already justified in chapter II.B. where it was explained that in general the length of a convoy should not change too much during the travel and therefore the speed of all vehicles within the convoy can be assumed to be identical.

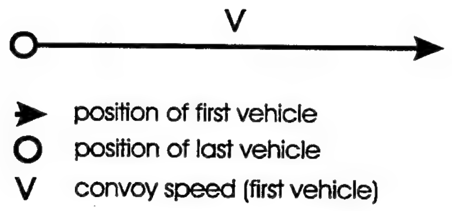

V
➤ position of first vehicle
O position of last vehicle
V convoy speed (first vehicle)

Figure 5: Convoy Model.

An important factor for the occupation of the route network is the length of each traveling convoy. This length for each convoy can be calculated easily by equation (2).

n	number of vehicles in a convoy,
d	distance between vehicles,
l	average length of a vehicle,
L	total convoy length

$$L = n \cdot (d + l) \qquad (2)$$

Realistic values for these parameters were given in Chapter II.D.

3. Event Points

An event model was used to simulate the movement of the ground forces. The location of the entities only have to be calculated when an event occurs and only those entities which are involved in the event enter the calculation. It is therefore important to predefine all events which can occur during such a simulation. Events happen at certain points along a route on which a convoy is traveling and influence the speed of the convoy in certain ways. They allow the convoy to travel faster, slower or stop. This means that it is important to calculate the time at which the leading vehicle of a convoy (convoy head) reaches one of these event points.

There are two different type of events points which interact with the convoy head.

A) STATIC EVENT POINT

A junction of a road indicates a new road section which might restrict the speed of the convoy.

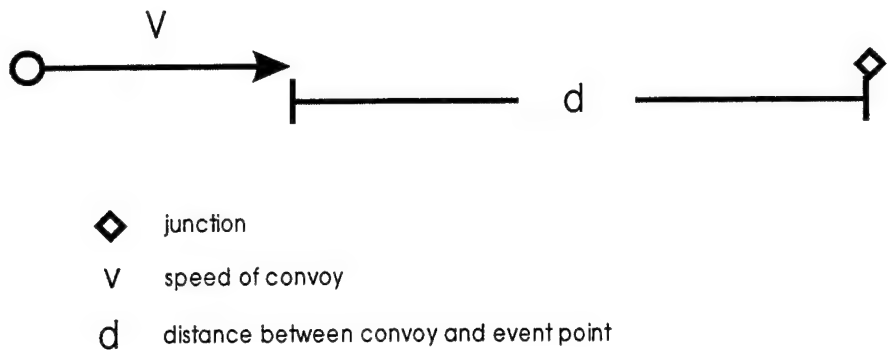


Figure 6: convoy moving towards a junction.

Figure 6 shows the determining parameters for such a situation. The time t until such a speed change is reached is easy to calculate using equation (3).

$$t = \frac{d}{v} \qquad (3)$$

B) DYNAMIC EVENT POINT

The second type of event point is a convoy traveling ahead of the observed convoy. This situation is shown in Figure 7.

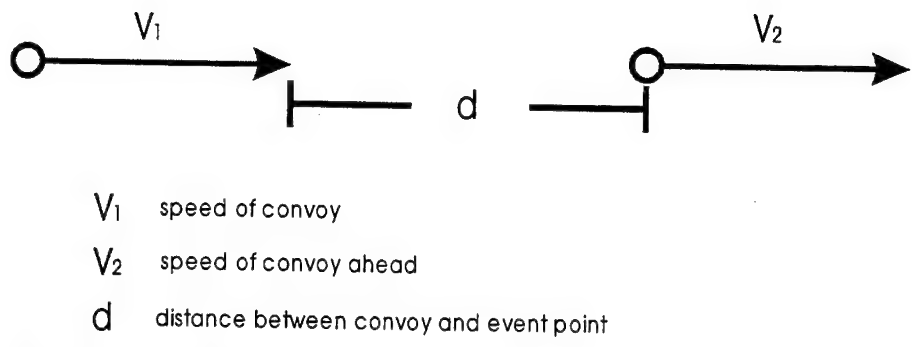


Figure 7: Convoy following another convoy.

This time the point for a possible speed change is moving at the same time as the convoy moves. It is important to note that the preceding convoy does not necessarily inflict a speed change on the following one. This is only the case if the speed v_2 is smaller than the speed v_1 of the following convoy. In this case the time t until the convoy reaches the speed change can be calculated with equation (4).

$$t = \frac{d}{v_1 - v_2} \qquad (4)$$

It is also important to note that the time until a convoy reaches the end of a preceding convoy has to be compared to the time until the convoy reaches the next junction on the road. Only the event with occurs earlier has to be handled by the simulation.

4. Blocking of road crossings

Road junctions where more than two roads are connected represent road junctions and crossings. Convoys can occupy a crossing in such a way so that other convoys cannot use the crossing to carry on their movement. An example of this situation is shown in figure 8. However, this is not always the case. Under certain circumstances a crossing can be used by two and even more convoys as shown in Figure 9. For modeling purpose it is assumed that convoys always drive on the right side of the road and the road is wide enough so opposing convoys can bypass each other.

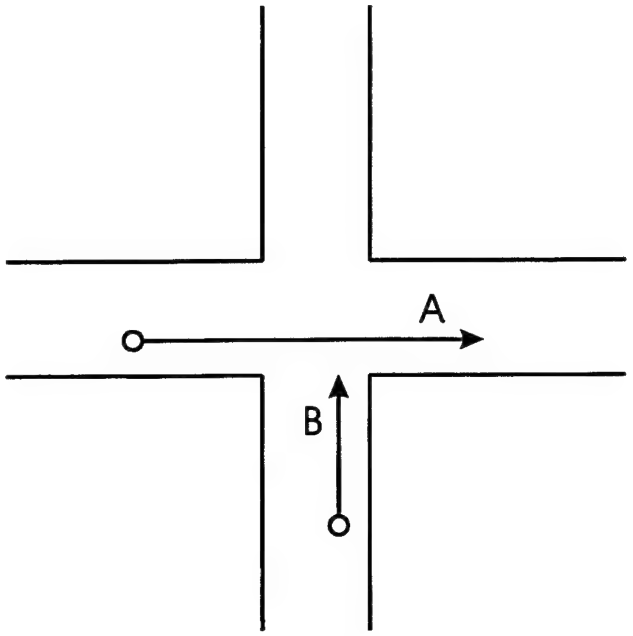


Figure 8: Convoy A blocking a intersection against convoy B.

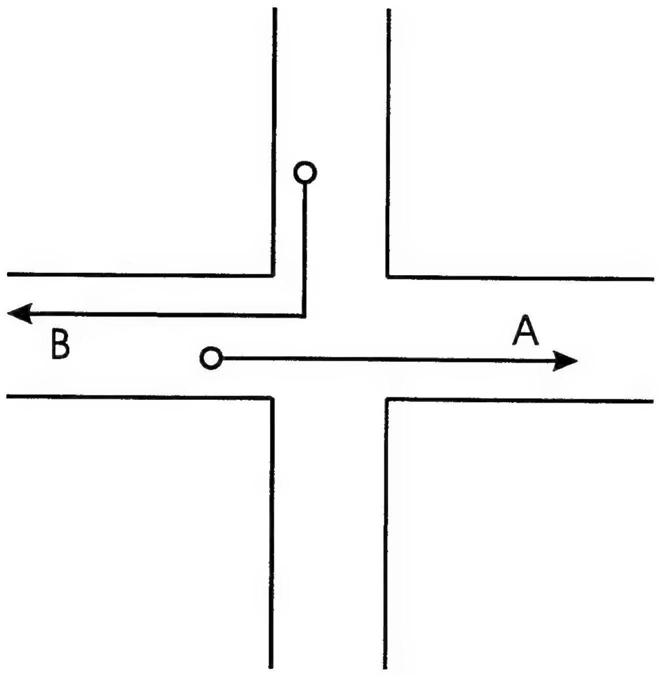


Figure 9: Convoy A and B operating through the same intersection without blocking.

In general, a nontrivial road junction might consist of three or more road sections which are connected to it as shown in Figure 10.

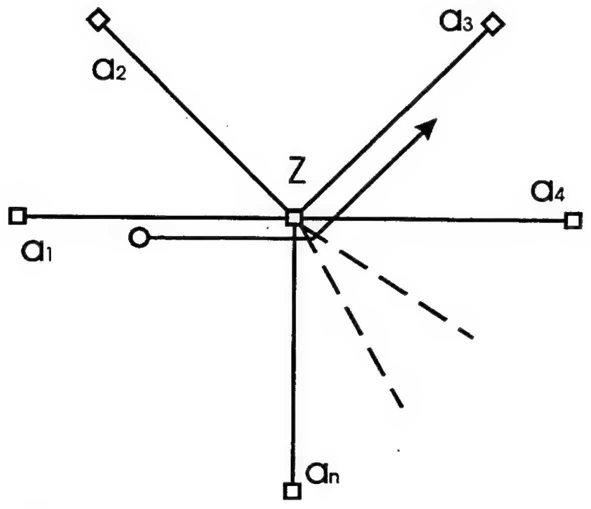


Figure 10: N road sections connecting to the same intersection.

In Figure 10 the intersection node is called Z and there are n road sections connecting to it. Here, a convoy is moving from node 1 to node 3. This movement through the intersection is not blocking it off completely for any other movements; the following movement of convoys would still be possible without any conflict:

- convoy from node 2 to node 1,
- convoy from node 3 to node 1 and
- convoy from node 3 to node 2.

All other movement possibilities are blocked by the convoy currently occupying the intersection.

This simple example visualizes the principles of movement of two convoys through the same intersection. In order to determine if an additional convoy can move into a intersection we have to compute the following:

Number the connected nodes to a intersection in clockwise order a_1, a_2, .. a_n. Without loss in generality let a_1 be the originating node from where a convoy is traveling and a_i the target node of the convoy. Therefore the movement from a_k to a_j is still possible, under the conditions: $k > j$, $k \leq i$ and $j \geq 1$.

These conditions enforce the normal traffic law on a road junction which can be assumed to be also valid under operational conditions.

5. Event model

In IVY each convoy is represented by the location of the convoy head and the location of the convoy tail. The simulation keeps track of the actual position of these two points and schedules events for them. The convoy head determines the speed of the convoy and the tail is adjusting all the time to this speed. Figure 11 shows the Event Graph (Buss 1995 and Buss 1996) for IVY.

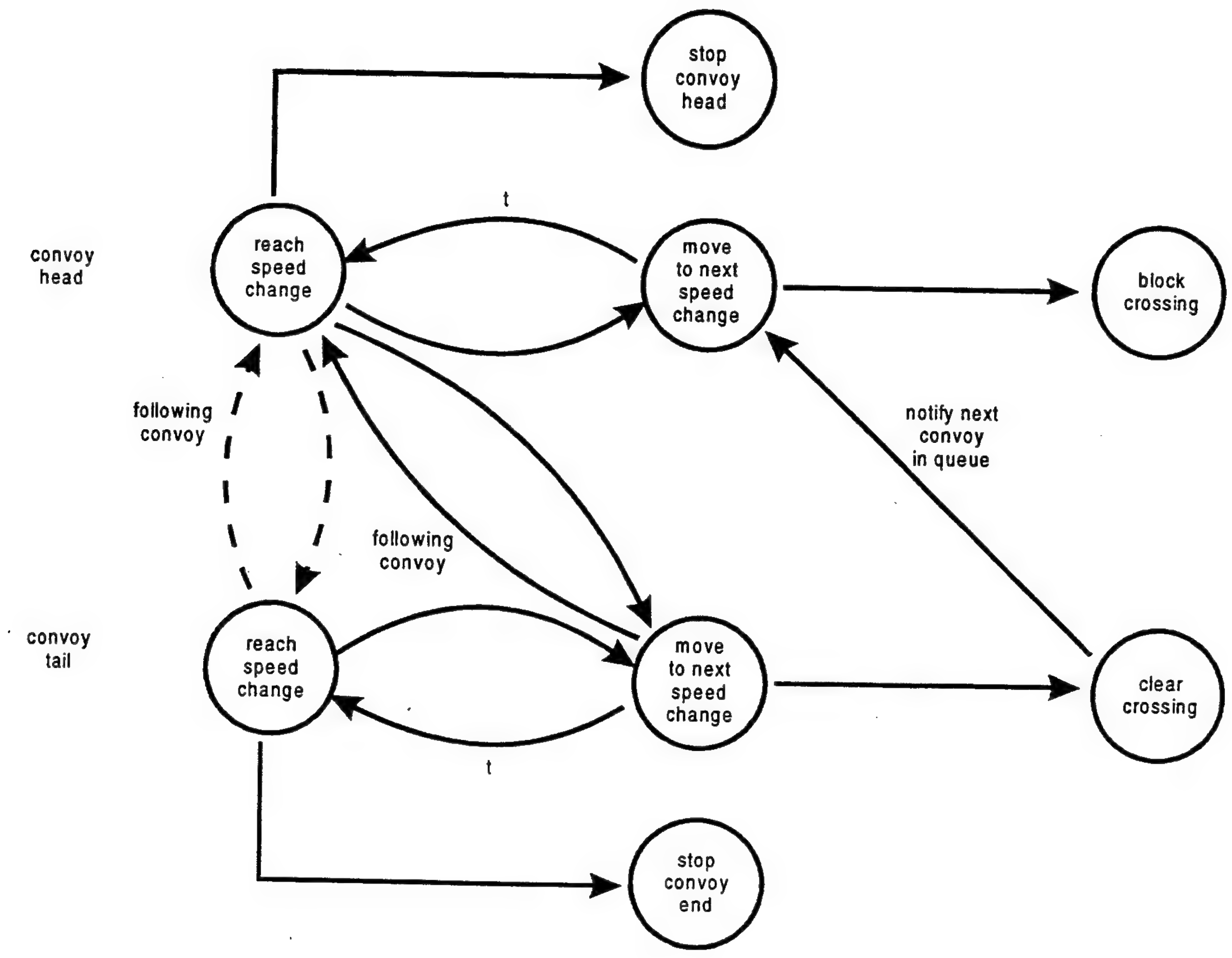


Figure 11: IVY event graph.

The convoy head moves from speed change to speed change on its given route. A speed change can be either caused by a junction which is the starting point for a new road section with its underlying speed restrictions or by a convoy traveling in front. In this case the location of the convoy end would be the reason for a speed change. In the event that this speed change is caused by a junction and the junction is free so the convoy can enter it, it automatically sets a block onto this junction. This does not block other convoys completely off from using this junction, but makes them aware that they have to check if the junction can be used in the desired movement direction. For blocking of intersections the rules developed in chapter II.B.4 are applied. The convoy head then calculates the time until it reaches the next speed change either a junction or another convoy tail. When the convoy head changes its speed it also notifies its tail about this change. The tail then has to calculate its new arrival time on the next junction. Upon arriving at a junction the convoy tail takes the block off this junction so other convoys can use it more freely. With every speed change the convoy tail notifies a possibly following convoy about this event.

In the event that there are no more junctions on the route of a convoy, the convoy head just stops at the last junction. The convoy tail in this case will stop at the point of the road where it got at this time in the simulation. Because of the fact that a convoy stopping on a road blocks this road to other convoys, it is a good idea to create at the end of a route a deployment area for the convoy where it can get off the main road.

This theoretical model of the movement simulation is implemented in Java as shown in Figure 12.

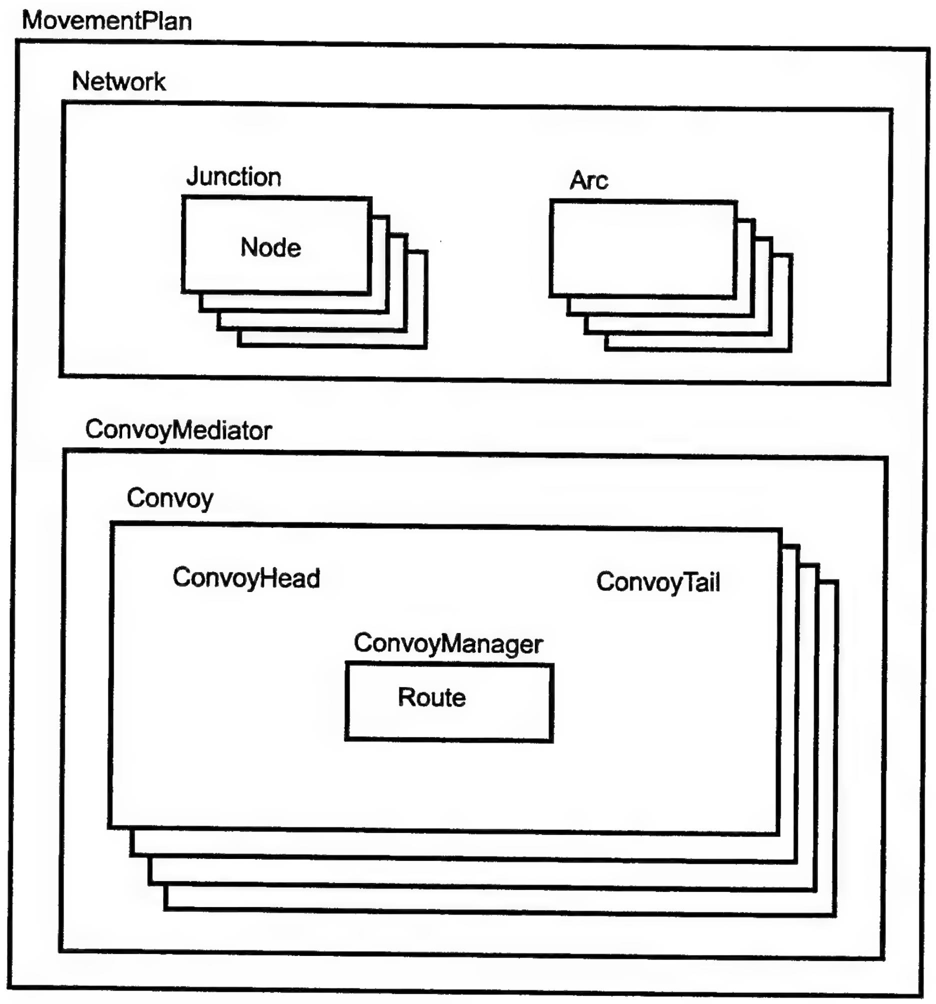


Figure 12: Component structure of IVY.

The movement plan holds information about the route network as well as about all the participating convoys. The route network exists of a number of Junctions which are connected with arcs. This forms the available routes for the simulation. Each arc has a variable number of properties such as speed, height, etc. restrictions. In the IVY movement model only speed restrictions are handled. This allows the calculation of the maximum speed a given convoy can travel along a road section.

All convoys are monitored by the convoy mediator. The mediator schedules events of one convoy running into another convoy's tail. On this event the convoy mediator then notifies the convoy about reaching the tail of another convoy and the following convoy adjusts its speed. The advantage is that the convoys do not need to have any knowledge about each other within the simulation. This implementation is very close to reality where a convoy head realizes another convoy traveling in front of it only when it actually reaches this other convoy's tail. Each convoy is determined by the convoy head and tail location. These two entities are simulated independently and only get triggered from each other by scheduled events, such as when the convoy head changes its speed. The "brain" of a convoy is the convoy manager. It holds the information about the route on which the convoy is ordered to travel and notifies the convoy head at each junction which direction to travel next.

IV. SAMPLE DATA

IVY is a pure movement simulation and has no means of optimizing a given movement scenario. Therefore all movements have to be determined in detail at the beginning of the simulation.

The available road network is determined by the BH Command Route Information map, dated December 5th, 1994. This map shows the available and observed route network which was used by UN forces and convoys. The displayed routes were modeled for the IVY movement simulation as described in chapter III.B. Speed restrictions were assigned to the individual routes; this influences the movement of convoys.

Figure 13 shows the route network which is available for movement of NATO forces.

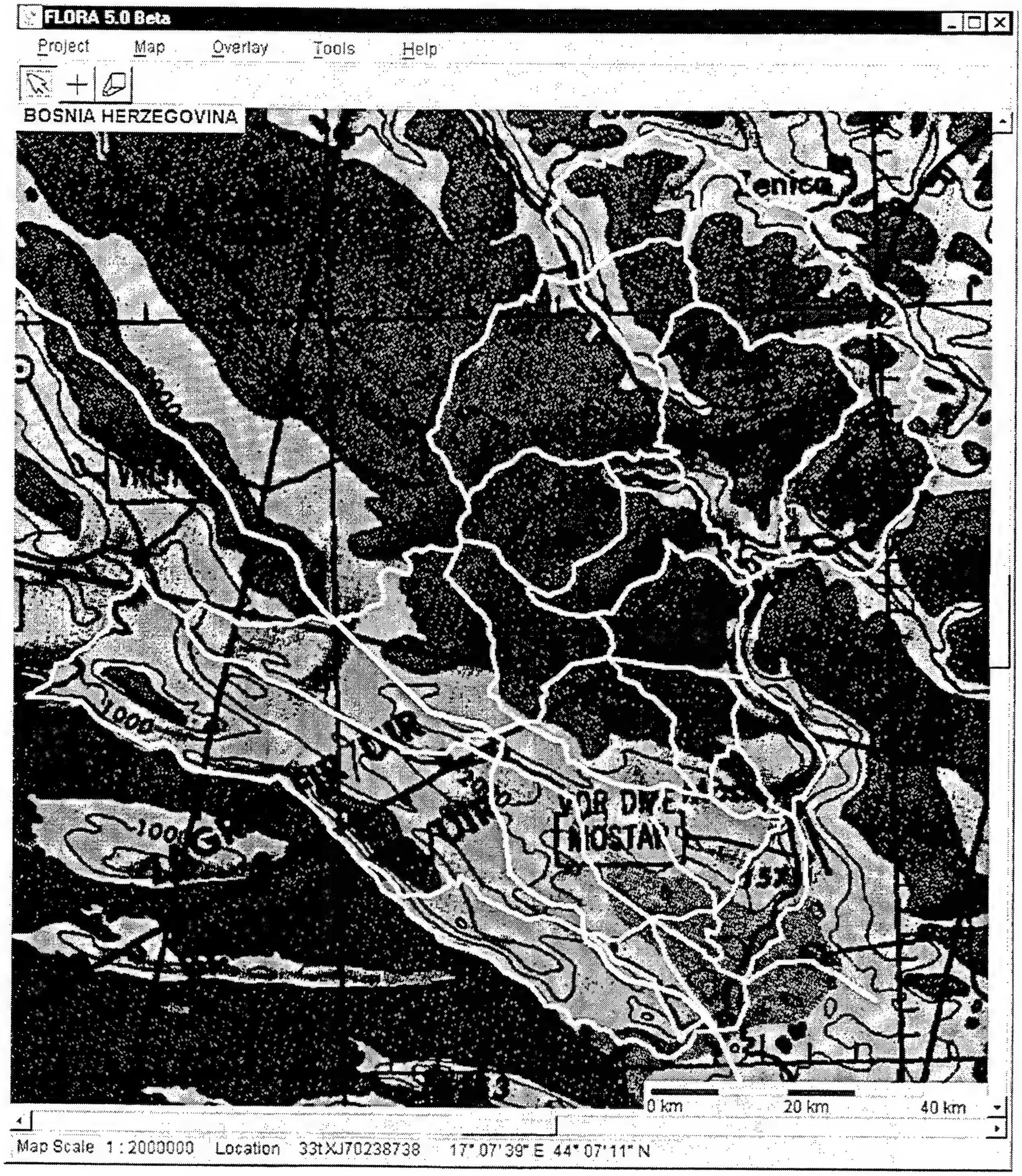


Figure 13: Route network modeled in IVY.

For the NATO forces one generic division is modeled. Figure 14 shows the structure of this division. The area of operation for this division is south-east of Sarajevo and therefore only the available route network between the Adriatic coast and Sarajevo is modeled.

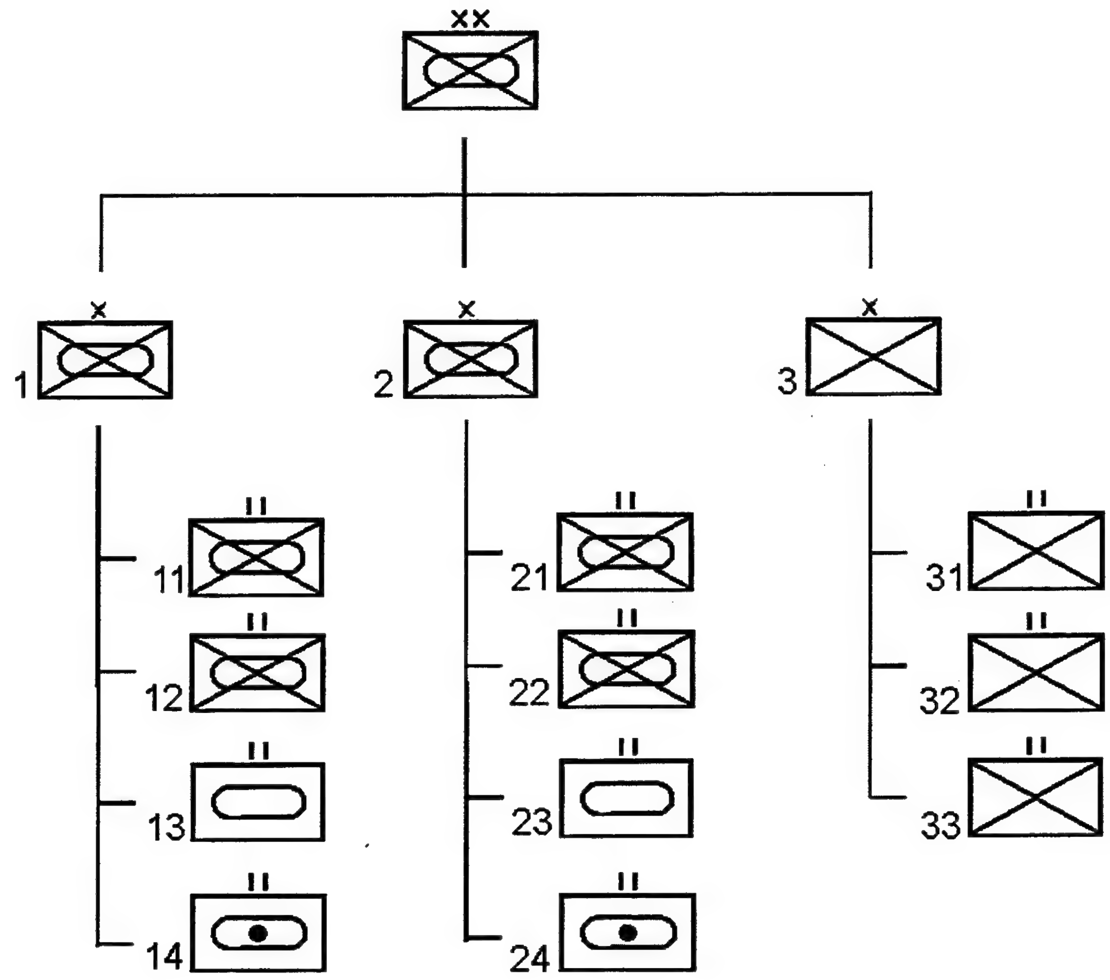


Figure 14: Sample division.

The mechanized division consists of two mechanized brigades and one infantry brigade. Figure 14 also shows the number of battalions which each of the brigades has under its command. Each of these battalions has between four and six companies and each company has between 20 and 50 vehicles. During the movement phase combat battalions normally stay together. Combat support battalions, however, quite often are split up and attached to combat battalions. In this case these units move together with the formations they are attached to. However, this does not have always to be the case and it depends on the threat assessment and future tasks of these units. For this simulation each combat battalion is moving as a formation together with some attached combat support troops. The movement objectives of the brigades are shown in Figure 15.

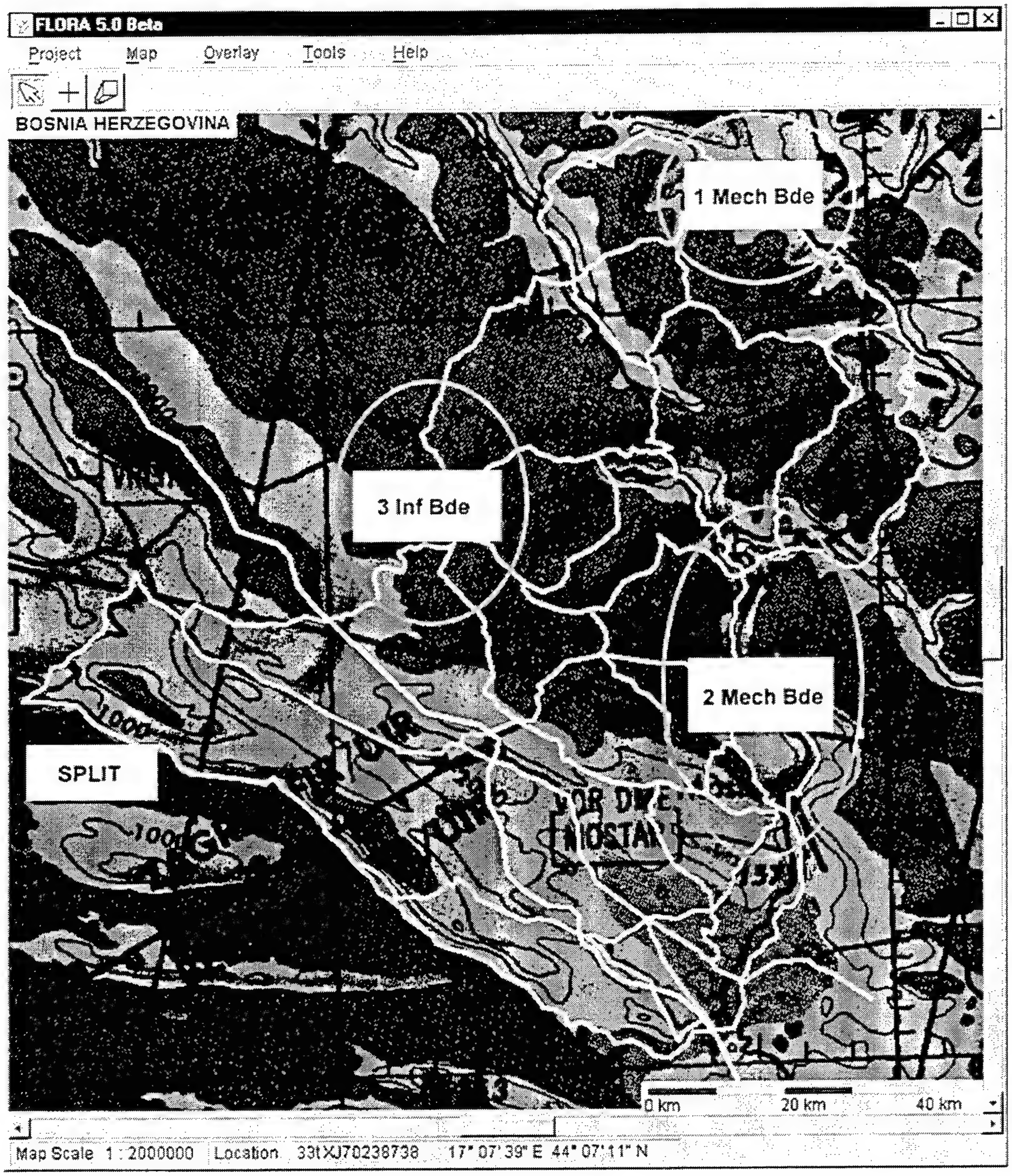


Figure 15: Movement objectives of the brigades.

The only available harbor which is capable of unloading a large number of units and equipment is at Split. Therefore, all formations have to start their movement from this location.

At the same time as NATO forces are trying to evacuate UN forces from BH it was expected that the fighting between the warring factions would become even more intensive. This would then result in large refugee movements wherever one party gains some success in taking over some territory. These refugee streams are represented in IVY as convoys which move along roads from inside BH to the Adriatic cost. Like military convoys, these occupy road sections and crossings and therefore conflict with the movement of NATO forces.

V. ANALYSIS OF RESULTS

A. COMPUTATIONAL EXPERIENCE

The development of the scenario data for the IVY momvenent model was done in two steps. The route network was digitised by using FLORA and a route builder component, which allowed digitise the routes seen on the map display. In a second step the movement plan for the convoys was produced in a similar maner. The complete setup process of the exercised scenario took less than two hours.

The software structure of the loosely coupled components proved to be a great advantage during the development and programming phase of the simulation. The development of a complex simulation which includes the movément of forces is not a trivial task. A pure text based output with hundreds of grid locations for moving units can only be debugged to a certain extend. One of the major problems during the development of the movement simulations was to synchronize the movement of the convoy head and tail of a convoy. The capability of visualizing the convoy location on a graphical display made it possible to discover errors in the simulation rapidly. The various zoom options of FLORA allowed the analyst to examine certain aspects of the displayed positions of the convoys in great detail. The movement simulation could not have been developed easily or debugged effectively without such a display tool. Due to the loosely coupling of software components it was possible to develop IVY as a pure simulation. IVY used the given software architecture to send the simulated convoy locations to FLORA for display.

B. SIMULATION OUTPUT

The start positions of all the entities in the movement simulation are shown in figure 16. All formations of the simulated division are starting from the area around Split.

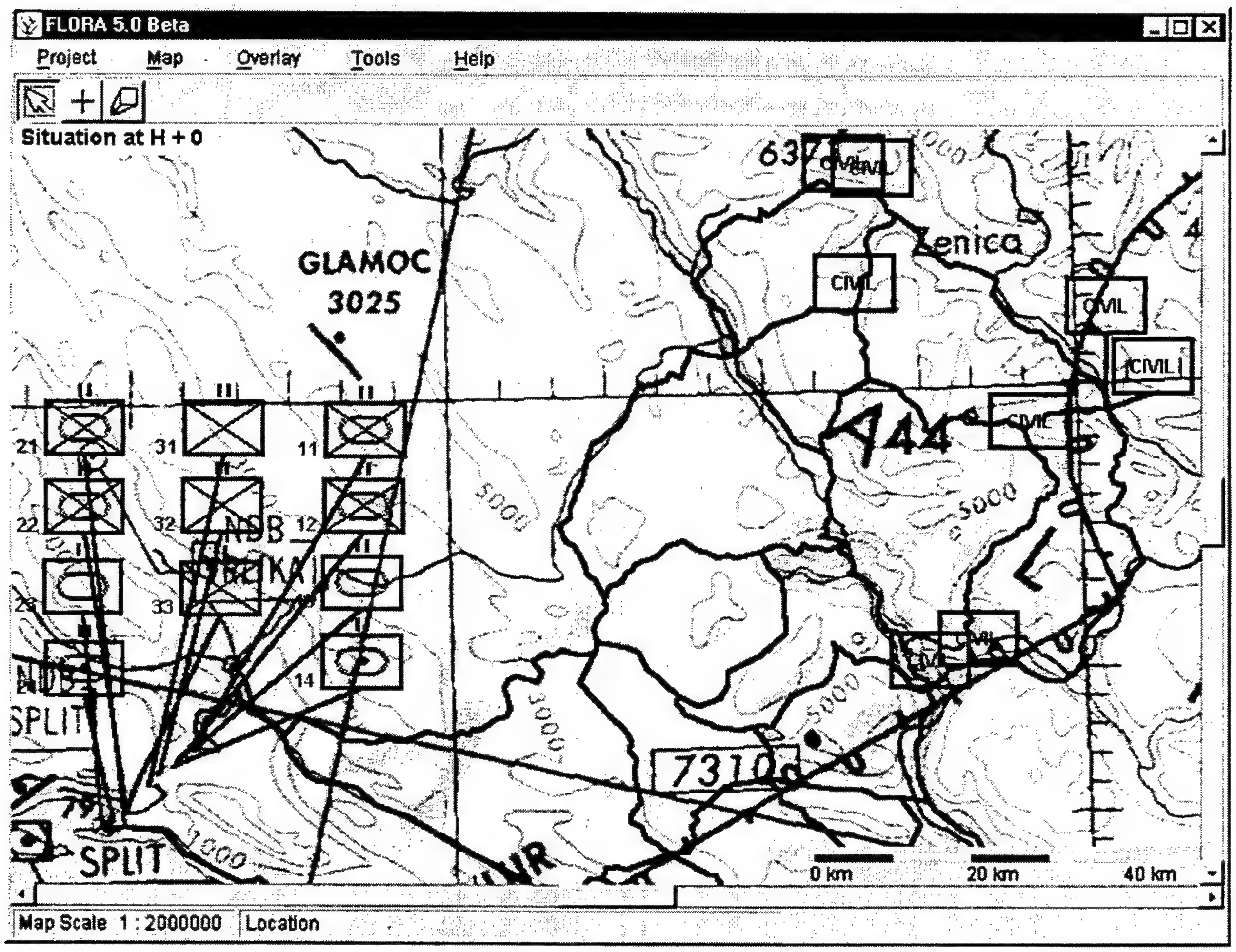


Figure 16: Start positions for simulation run.

The combination of simulation and graphical display made problem areas in the movement plan immediately visible. Figure 17 shows a movement conflict between two units trying to access the same intersection. While the 33rd Infantry Battalion is carrying on its movement, the 31st Infantry Battalion has to wait until the intersections is clear.

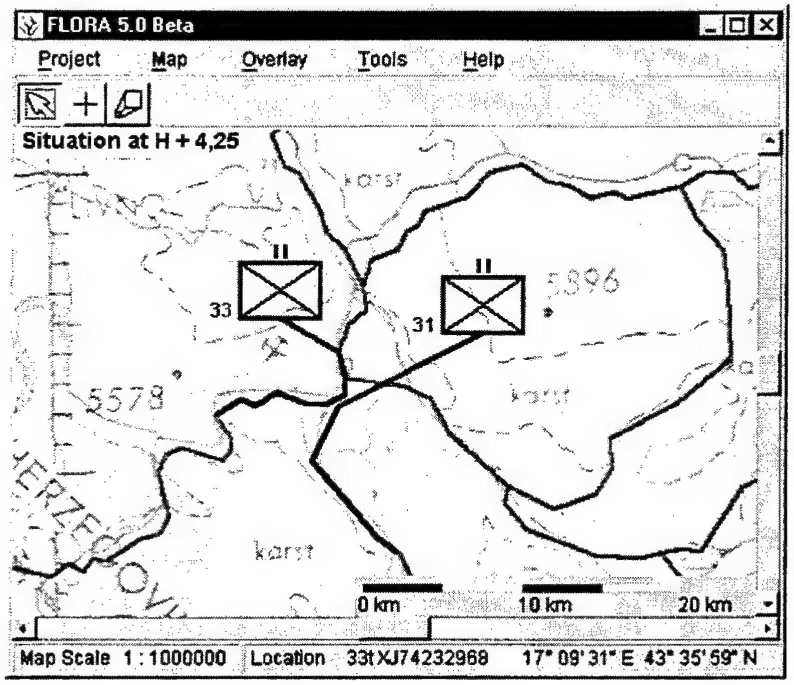


Figure 17: Movement conflict between two convoys.

Figure 18 shows a situation where military convoys are bypassing opposing refugee convoys in opposite directions. The movement model IVY allows bypassing opposing convoys, but convoys traveling in the same direction can not do so. Therefore the right refugee convoy is queuing behind the left one. The 11th Mechanized Infantry Battalion nearly has passed both refugee convoys and the 12th Mechanized Infantry Battalion is still in front of them.

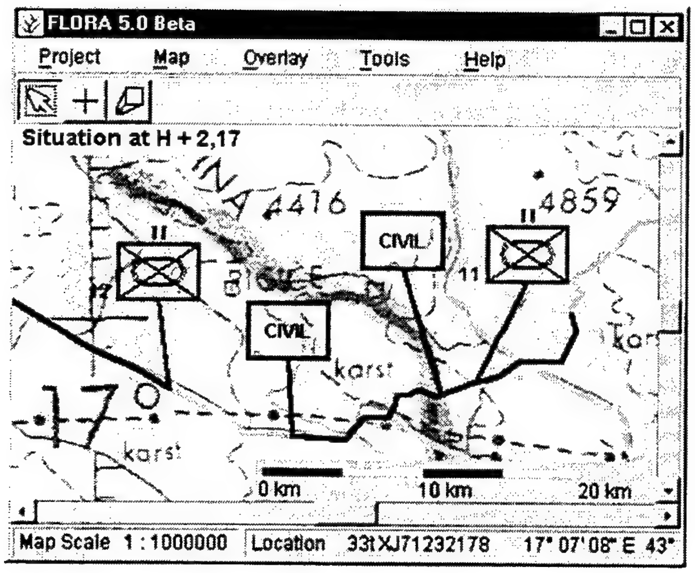


Figure 18: Occupation of roads by military and civilian convoys.

Figure 19 shows the final state of the movement simulation 8.5 hours after the start of the operation. All NATO forces reached their movement objectives and can now secure these areas.

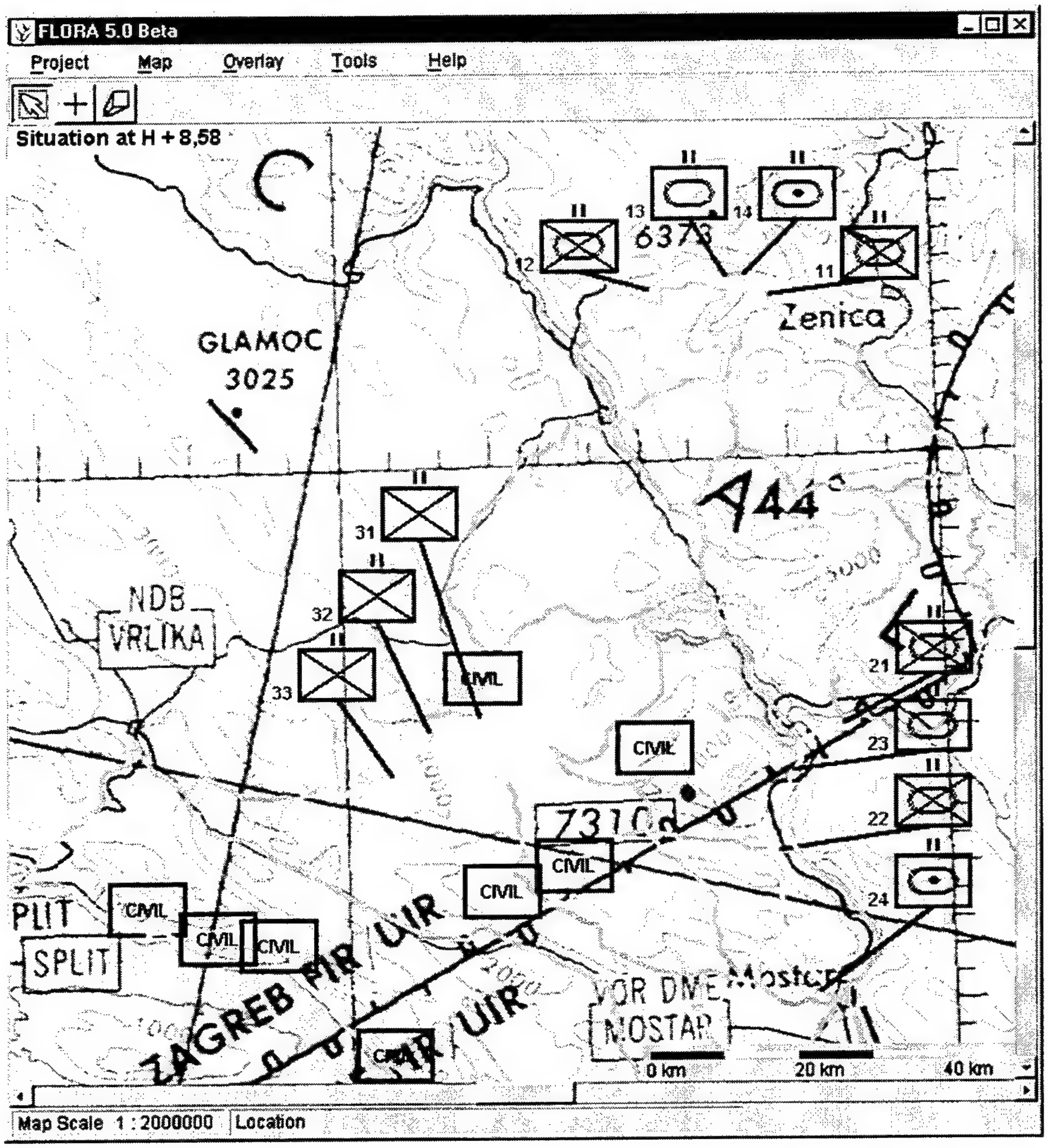


Figure 19: Final graphical output for the exercised movement plan.

C. COMPARISON WITH THE SITUATION IN 1995

In 1995 the HQ ARRC had only a very limited set of tools available for the planning of ground movement, such as a spreadsheet and a hard-to-program movement simulation. The movement simulation IVY developed in this thesis allows a flexible evaluation of movement plans by visualizing them on a geographical display. Movement conflicts like blocking of convoy at road junctions and bottlenecks can easily be spotted. It is then up to the planning staff to propose a new movement plan which can be evaluated in the same way.

Visualizing the results of a movement plan is also a good way to build up confidence in the software. IVY is thus a very powerful tool for briefing the decision maker who will be able to understand the implications of a proposed movement plan much easier on a geographical display than on a timetable with endless columns of grid coordinates and times.

A further benefit of visualizing the simulated movement is that it is now very easy to spot possible threats to a plan, which might not be obvious in a pure simulation, because they do not affect the particular run.

VI. CONCLUSIONS

The idea of loosely coupled software components allows an analyst to select the modules that are required for a specific task and combine them, so they can interact with each other. In this way every component can be written to execute only one specific task. The combination will then take the benefits of all available components and create a powerful analysis tool.

This thesis develops the software architecture THISTLE for a set of loosely coupled components which need a way of interacting with each other but still have to be capable of being executed in a stand alone mode. Because all communication within the THISTLE environment is done through the Message Center, the individual components have no knowledge about the functionality of other components with which they are communicating. This allows the replacement of components as long as the newly introduced one provides a similar functionality.

To prove the concept of THISTLE, the map display component FLORA and the movement simulation IVY were developed. Both can be executed as stand alone applications. FLORA is capable of displaying maps and tactical symbols in a geographical referenced environment. It also allows the user to select a set of different zoom levels and to import background maps from various sources. FLORA also allows the user to load and display a unlimited number of overlays simultaneously. For better visualization, the background maps can be faded out so the tactical symbols are easier to see. IVY, as a movement model for ground forces, simulates the movement of convoys along a given route network. During the simulation run IVY produces a list of locations which represent the position of each convoy at any given time. The THISTLE environment allows the interaction of FLORA and IVY, so that the analyst can view the result of the movement simulation on a map background and can therefore easily visualize the shortcomings and threats to the simulated movement plan.

The software architecture THISTLE of loosely coupled components provides a very powerful programming environment, which allows the user to easily combine different components for a specific task. The map display tool is one of the cornerstones of the system, which can be used for many different geographical related simulations and models. IVY provides a useful simulation for the movement of ground forces in Operations Other Than War, where the timing and deconfliction of convoys is essential. The combination of both shows how two components can benefit of each others capabilities and therefore create a powerful tool for planning and analysis purposes.

APPENDIX A: GRAPHICAL USER INTERFACE

The graphical user interface is built to give the user maximum flexibility for viewing an ongoing simulation and its results. Whenever sensible, shortcuts have been defined to allow fast switching between different graphical views of the displayed data. To allow easy customization of the individual module, initialization files ("ini" files) are associated with each module. The layout of these "ini" files follows the rules of the Microsoft "ini" files:

- group headers are included in brackets (e.g. [DISPLAY]),
- variable names are followed by an "=" sign without a space in between,
- the value of the variables follows immediately the "=" sign without an space in front,
- comment lines start with a ";" as the first character. These lines are only useful for the user, but are ignored by the software.

All "ini" files are ASCII text file which can be edited by any text editor, such as Microsoft Notepad. Within the THISTLE software package there is a default "ini" file for each module. The user has the option to provide a customized "ini" file in the startup directory, which will overwrite the default "ini" file.

A. THISTLE TOOL BAR

The THISTLE toolbar, as shown in Figure 20, is a convenient way of launching the individual software tools within the same Java virtual machine.

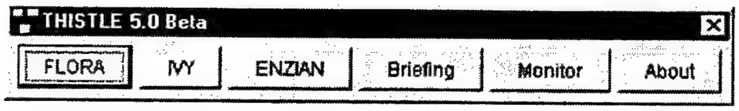


Figure 20: THISTLE tool bar.

Running all software tools within the same virtual machine is necessary for the exchanging messages between the individual tools and models. The buttons on the tool bar start the modules, which are specified in the thistle.ini file.

New software modules can be registered before starting up the THISTLE tool bar in the "ini" file. An example of a thistle.ini file that illustrates the format is given in Figure 21.

```
;THISTLE ini file

[MODULE0]
NAME=FLORA
CLASS=thistle.flora.FloraMain

[MODULE1]
NAME=IVY
CLASS=thistle.ivy.IvyMain

[MODULE2]
NAME=ENZIAN
CLASS=thistle.enzian.EnzianMain

[MODULE3]
NAME=Briefing
CLASS=thistle.briefing.BriefMain

[MODULE4]
NAME=Monitor
CLASS=thistle.message.Monitor
```

Figure 21: Example of Thistle.ini file.

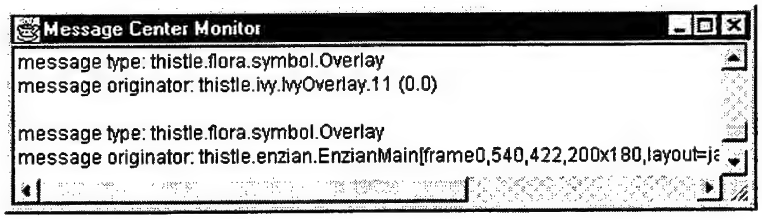
In theory there is no limit on the number of modules which can be registered with the THISTLE tool bar.

B. MESSAGE CENTER

The heart of the THISTLE system is the message center, which provides the loosely coupling of the individual software tools. It is responsible for broadcasting messages to all registered software tools. The message center itself is static and does not have to be explicitly instantiated. It is automatically available the first time it is invoked by any module. There is no visual representation in the form of a window on the screen for the message center. A more detailed technical description of the message center can be found in Appendix B.

C. MESSAGE CENTER MONITOR

The message center monitor provides a view on the activities which happen inside the message center. It is important to note that message center monitor is not a part of the message center. Rather, it is an individual tool which acts as an observer to the message center and monitors its behavior. This tool is useful for debugging and visualizing the message traffic. The "Monitor" button on the THISTLE toolbar starts the message center monitor. An example is shown in Figure 22.



```
message type: thistle.flora.symbol.Overlay
message originator: thistle.ivy.IvyOverlay.11 (0.0)

message type: thistle.flora.symbol.Overlay
message originator: thistle.enzian.EnzianMain[frame0,540,422,200x180,layout=ja
```

Figure 22: Message Center Monitor window.

Each message is displayed with two lines in the message center monitor. The message type refers to the class of the message object. Software modules inspect this property and decide, based on its value, whether this message can be used for further action. FLORA, for example, only listens to objects of the class “thistle.flora.symbol.Overlay”. The reference to the message originator is useful for identifying from which module a certain message was send. It contains a reference to originating software module.

D. GEOGRAPHICAL USER INTERFACE - FLORA

One of the most important software modules within Project THISTLE is the map display tool FLORA. This module shows geographically related data on a scaleable map display. It also converts the cursor position on the map display into real world coordinates like Latitude/Longitude or Universal Transverse Mercator (UTM).

On start-up, FLORA registers automatically with the message center. Therefore it will be notified of all new messages, whereupon it will examine the message for content and, if possible, it will display it on the current map display. Currently FLORA can only display messages if they are send in CDTF format and wrapped in the Overlay class (object of type thistle.flora.symbol.Overlay).

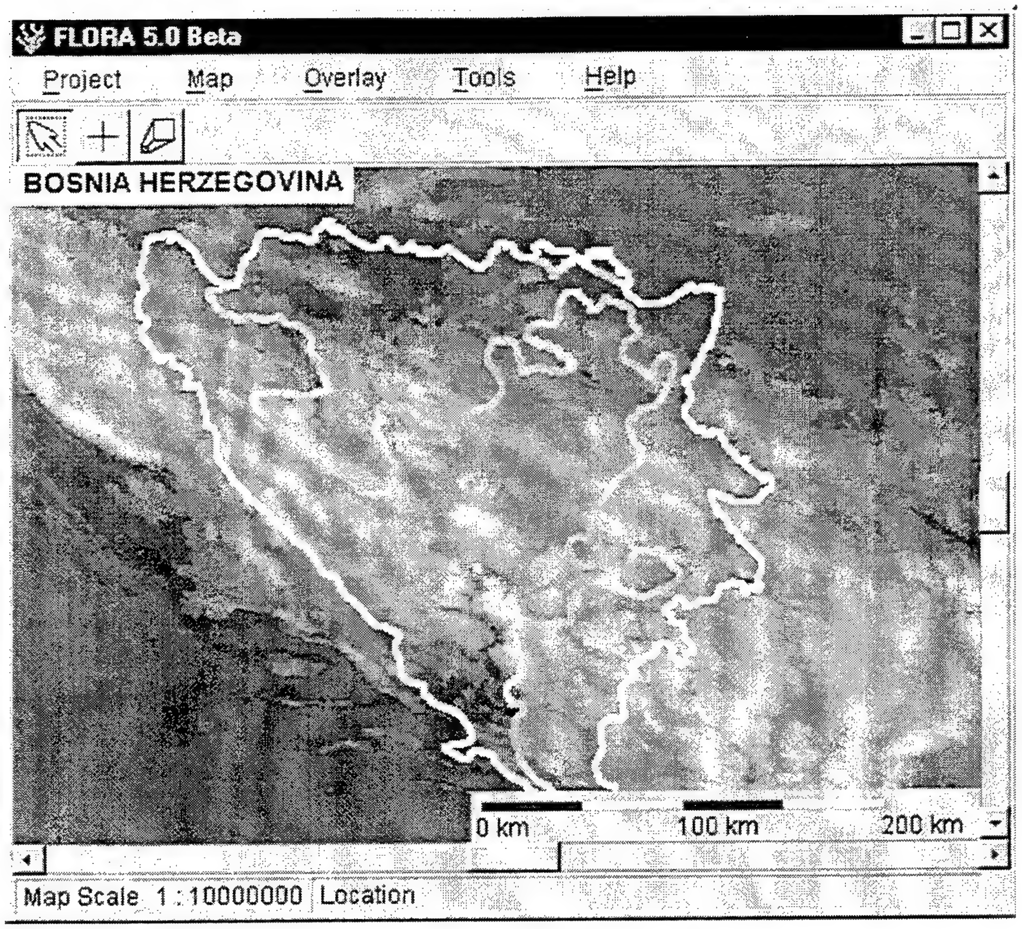


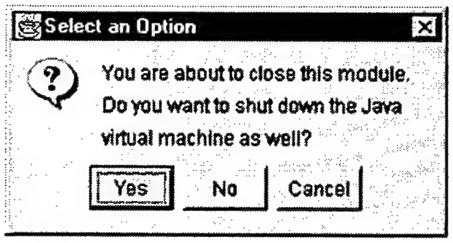
Figure 23: FLORA map display with overlays.

1. USER INTERFACE

The FLORA user interface is split up into a map display area, a menu bar, a pop-up menu, a tool bar and a foot bar. The map display area and the foot bar display information to the user about the current map and scenario. The menu bar and the tool bar provide the functionally for manipulating the map display.

In general the menu bar provides access to all functions which are implemented in FLORA. For faster user interaction it is convenient for the skilled user to use the defined speed buttons and hot keys on the tool bar.

The menu item 'Exit' within the menu group 'Project' will close the FLORA screen and exit the module. When selected it will give the user the option to close down the Java virtual machine together with the FLORA module. To prevent an unintentional shut down of FLORA and a resulting loss of data, the user has to confirm this action on a pop-up window (see Figure 24).



You are about to close this module. Do you want to shut down the Java virtual machine as well?

Figure 24: Popup window for closing the Java virtual machine together with the FLORA.

It is important to realize that by shutting down the virtual machine all currently executing THISTLE tools will be terminated. This might not be always the intention of the user. Therefore by selecting the 'No' option only the FLORA module will be exited. The hot key for closing FLORA is ALT+F4.

The menu group 'Map' offers options for manipulating the map display. The 'zoom' item allows the user to 'zoom in' and 'zoom out' relative to the current map display. For these zooming operations FLORA will try to make the center of the current display the center of the new display. This is especially important for zooming in operations, because otherwise the new display might not be as intended by the user. There are two hotkeys assigned to this operation. [CTRL+PAGE_DOWN] is the short cut to zooming in and [CTRL+PAGE_UP] implements zooming out. To specify a certain zoom in center on a map the user can right click on the map display and bring up a pop up menu from where the zooming operation can be selected. In this case FLORA will use the selected point on the map as the zoom center. During some simulations or briefings it might be desired to zoom in and out at certain map points just with a mouse click on the map. To accomplish this FLORA mode has to be changed into 'Zoom Mode' by pressing the zoom in/out button on the tool bar. In this mode every mouse click on the map will result in a zooming operation.

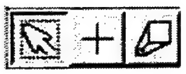


Figure 25: The interrogate, edit and zoom in/out button on the tool bar.

A left mouse click will now correspond to a 'zoom in' operation at with the center point at the selected point on the map, while a right mouse click will result in a 'zoom out' operation. This is a very powerful way for changing the view on the current map display and for looking at a constantly changing operation.

Scrolling the map display can be done in a variety of ways. The menu group 'Map' offers a sub menu scroll from where the map can be scrolled up, down, left and right. Using these options will be probably too slow and it is not recommended in general. However, it might be useful for fine-tuning of the map display and to get FLORA to display a certain area of the map. This operation can also be accomplished by holding the CTRL key down and using the arrow keys for scrolling the map in the desired direction. The most powerful option is to select the interrogate mode on the tool bar and to grab (click and hold down the mouse key) the map and drag it to the desired position.

Sometimes the colors of the map might be too bright and thus prevent the user from seeing the currently displayed tactical situation. FLORA provides an option to either darken or lighten the background map and therefore to make the tactical symbols stand out. The menu item for this operation is also in the menu group 'Map' under the sub-menu 'Brightness'. The hot key for darkening the map is CTRL+NumPad MINUS and for lightening CTRL+NumPad PLUS.

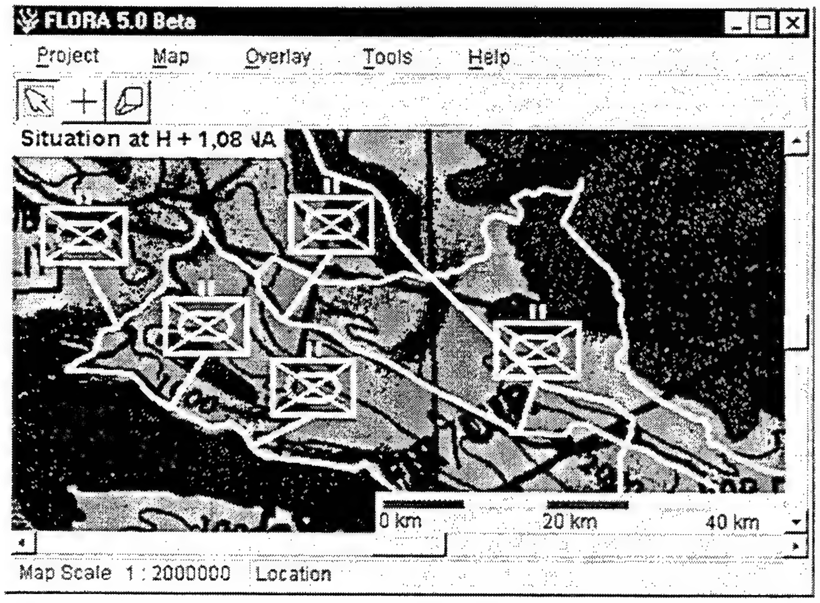


Figure 26: Darkend map display of FLORA.

For easy referencing the FLORA display has the option to display the scale bar in the lower right corner of the display. This allows the user to better visualize distances on the map. It might be desirable to turn this scale bar off in certain situations. Selecting 'Scale Bar' in the 'Map' group of the menu bar allows the user to do so.

A powerful feature of FLORA is its capability to display tactical symbols on top of a map display. As in the real world, this is done by putting overlays onto the map.

Overlays can be loaded manually into the FLORA display. The menu group 'Overlay' offers the menu item 'Add overlay' for selecting an overlay. This overlay will then be loaded into FLORA and displayed together with all the previously loaded overlays. The name of the last loaded Overlay will be displayed in the left top corner of the map display. The option 'Remove all Overlays' will remove all overlays from the map display. Note that FLORA will also automatically load overlays if it receives them from other software modules via the message center. A more detailed technical description of this feature can be found in Appendix B.

FLORA also allows the user to select a location on the map display and to broadcast the grid location via the message center. To enable this feature, FLORA needs to be in the edit mode. This can be achieved by pressing the edit button on the tool bar. In this mode every mouse click on the map is converted into a coordinate and send to the message center for further distribution. This is a very powerful feature, which provides a mean of communication between FLORA and other software tools without direct interaction.

2. OVERLAYS

Overlays are used in real life to display information on a map without actually drawing on the map. The same idea is used with the FLORA module. The tactical information can be summarized on an overlay and then can be loaded by Flora and displayed on a map background. One of the advantages of FLORA overlays is that they re-scale with the map scale. There are two ways to load overlays into FLORA. As shown in the previous section, they can either be loaded manually by using the options from the menu bar, or they can be loaded automatically from other software modules by sending them via the message center.

3. FLORA.INI FILE

The "ini" file for FLORA (Flora.ini) allows the user to define the start parameters for this software module. This is useful when a certain map display and focus is desired on each start up. It allows the user to have the same view of the map after each start of FLORA and the user saves the time of adjusting the map display to his/her own needs each time. It is also important to note that not all of the values specified in the "ini" file can be changed during runtime. The following section explains the variables which can be adjusted.

```
[DISPLAY]
```

In this section the FLORA window is defined.

```
NOTILES=5
```

This is the number of map tiles in the x and y direction which are loaded for displaying the map. Any number from 2 to Integer.POSITIVE_INFINITY is possible. However, choosing this number too high will result in a loss of performance for updating the screen display. A number too low will actually result in a smaller displayed map with a gray border around it. The default value is 4.

```
RULER=visible
```

This specifies whether the scale bar for the map scale is visible at start time. The user can change this parameter during run time.

```
SCALE=100000
MAPTYPE=PAPER
CENTER_UTM=33t YJ 07000 70000
```

These values define the initial map scale and type to be loaded as well as the initial viewpoint. The viewpoint is defined with the bottom left corner map tile.

```
[MAPS]
```

Under this header the meta information for the available maps is stored.

```
PROJECT_HEADER=map/IFOR.hdr
```

This defines the path to the header file of the map area. This header file does not have to be kept together with the map data. The format of the header file is explained in the section about the map database

```
[OVERLAY]
```

This section specifies the initial values for displaying overlays on top of the map display.

```
BASE=overlay/
```

The default directory for overlay files. Note that there seems to be a bug in the current implementation of the Java virtual machine for Windows 95: it is not possible to specify the default directory in the file loader window. This feature is functional on a Windows NT platform (Chan/Lee 1997).

```
MAXSYMBOLSIZE=2
MINSYMBOLSIZE=.5
```

The maximal and minimal allowed symbol size scaling for tactical symbols on the screen.

```
DEFAULTSYMBOLSIZE=1
```

This variable defines the default symbol size scale for map symbols, which are not specified in size on the overlay. A symbol for which a scale factor of 1 is given will have the default size of 50 pixels wide and 34 pixels high.

```
DEFAULT0=overlay/BosniaBorder.cdtf
```

This variable defines the overlay which will be loaded and displayed at start time. By using the DEFAULT0, DEFAULT1, etc. variables an infinite number of additional overlays can be specified. Note that these variables have to exist without numbering gaps in order for the software to load them all.

4. MAPS

FLORA uses pre-calculated maps in order to shorten the loading times of maps. Because FLORA displays the maps in two dimensions, while the earth is a three dimensional sphere, the size of the map area is restricted to 1000 by 1000 km. This assures that the accuracy of the grid conversion is still below an error of 10 meter on a

1:100,000 map scale. For every zoom level a set of maps has to be prepared. The locations of all pre-calculated maps have to be defined in a project header file. The format of this file is discussed at the end of this section.

The maps are stored in GIF format which is a compressed picture format which can to store pictures with 256 colors without loss of information due to the compression algorithm. This file format was developed for viewing of images on web pages on the Internet. Compressing the maps into GIF format has the advantage of reducing the size of the original BMP files by a factor of three to four. GIF is a proprietary format, with the copyright held by CompuServe. Future developments of FLORA will add additional formats, such as JPG or PNG. PNG is an open source picture format with no copyright limitations. Currently there is no support for PNG images in JDK 1.2 (Weber 1998).

For fast access and easy referencing the map data is divided into a number of small map tiles. All map tiles of one type and scale (e.g. Scale 1:100,000, scanned paper maps) have to be kept in the same directory (location). Each map tile must be 100 by 100 pixel resolution. The file names for the map tiles consist of a eight digit number where the first four digits represent its location on a grid in horizontal direction and the last four digits the location in the vertical direction. The numbering schema for the map files is easily seen in Figure 27.

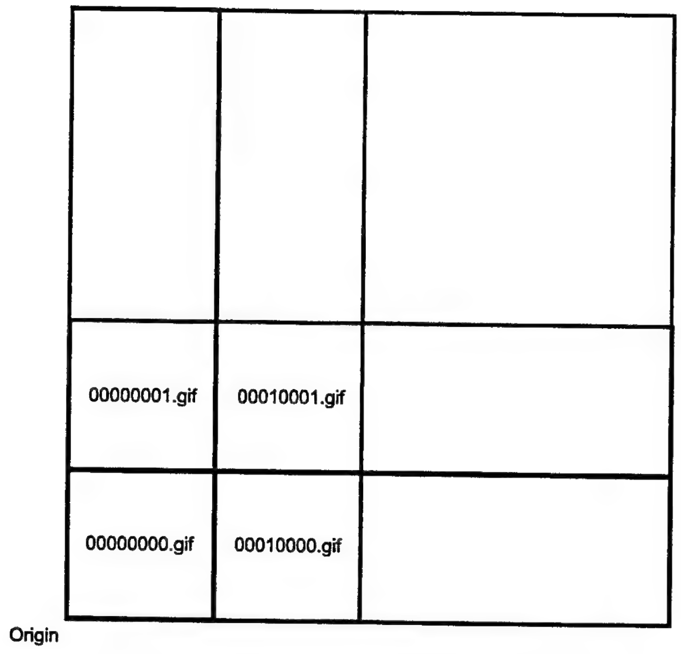


Figure 27: Organizational structure of map database.

The map project header file provides geographic information about the pre-calculated maps tiles. It also specifies where the available maps are located within the system. The center of the map area is given by the variables:

```
CENTER_LONGITUDE=17.971429
CENTER_LATITUDE=44.200000
```

Note that the project center can only be specified in longitude and latitude with decimal numbers. It is also required to specify the total map area in km. In general the project area should be square.

```
X_SIZE=1000
Y_SIZE=1000
```

The locations of different pre-calculated maps have to be specified under acending group names [MAP0], [MAP1], etc.. For each map the name, the type of the map source, the scale, the location, and the tile size have to be specified. Note that these entries are not optional.

```
NAME=ADRG
TYPE=PAPER
SCALE=500000
LOCATION=map/500000/paper/
TILESIZE=100
```

```
[GENERAL]
PROJECTNAME=IFOR 1995/1996
CENTER_LONGITUDE=17.971429
CENTER_LATITUDE=44.200000
X_SIZE=1000
Y_SIZE=1000

[MAP0]
NAME=ADRG
TYPE=PAPER
SCALE=50000
LOCATION=map/50000/paper/
TILESIZE=100

[MAP1]
NAME=ASRG
TYPE=PAPER
SCALE=100000
LOCATION=map/100000/paper/
TILESIZE=100
```

Figure 28: Example of the map project header file.

E. MOVEMENT SIMULATION – IVY

IVY simulates the movement of ground forces in an event step simulation. The control buttons for running the simulation are shown in Figure 29.

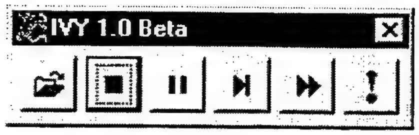


Figure 29: IVY module button bar.

This button allows the user to load a new movement plan into the movement simulation module IVY. IVY will automatically reinitialize all entries and be ready for running the simulation.

There are two modes in which the simulation can be executed: the normal event step mode or a pseudo time-step mode. The event step mode will run the simulation until all events are handled and the event list is empty. In contrast to this the pseudo time step mode will force the simulation to update the locations of all entities after certain time intervals and to display the current situation. This is done by sending an overlay with the route network and all convoy locations to the message center. From there it can be picked up by a geographical display tool such a FLORA and displayed on a map background.

The “start time step” button starts the simulation in time-step mode. The simulation can be pause at any point in time and after examining the display the simulation can be restarted by pressing this button again.

The “pause” button temporarily stops (pauses) a running simulation. It will stop the simulation, but not reinitialize the entities. This might be useful for examining certain aspects of an on-going simulation.

⏭ After the first run of a simulation it might be desired to run the rest of the simulation in the background without updating the screen display every time interval. This button will remove the location updating option from the simulation and run the simulation until all scheduled events are handled and the event list is empty.

■ The "stop" button is a convenient way of reinitializing the simulation at any given point in time. Note that all previously achieved simulation results will be reset and therefore lost.

! Pressing this button will force IVY to display some information about the author and a point of contact.

F. BRIEFING TOOL

The briefing module allows the user to load a number of overlay files in CDTF format and to display them in sequential order. The following figure shows the control window of the briefing module.

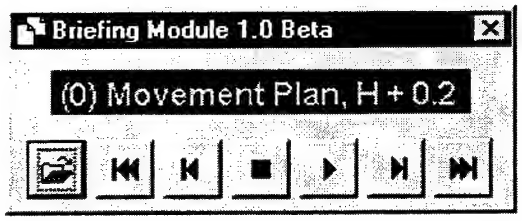


Figure 30: Briefing module button bar.

The briefing tool window shows a button bar which allows the user to load and display a set of overlays on a map display module and displays a status field which shows the name of the currently loaded overlay.

The "load slide show" button brings up a file input dialog window, which allows the user to specify the overlay summary file. This file specifies which overlays are to be loaded in the slide show and in which order they are displayed. The format of the overlay summary file is described at the end of this section. After successfully loading a set of overlays the title of the first overlay is displayed in the status field for the slide show.

The "single step" buttons allow the user to step through the slide show manually by displaying the next slide. The slides are kept within the briefing tool in a fixed order and it is possible at any point in the slide show to step forward and backward in this order.

The 'display first overlay' and 'display last overlay' buttons are convenient shortcuts for moving to the beginning and end of a briefing show.

The "auto-run" button steps through a slide show automatically by updating the displayed overlay every 2 seconds. This is a useful way of replaying a simulation for which a series of overlays are stored in CDTF format. The automatic display of overlays can be halted at any point by pressing the "stop" button.

■ The “stop” button stops an automatic running slide show at any point. The current slide is then shown on the FLORA display. This allows the user to examine the displayed tactical situation on the FLORA display. It is possible at this point to move manually forward and backward in the slide show.

The summary of the briefing overlays has to be stored in a *.show file. This file contains a list of the overlays which are to be displayed and in which order. In theory there is no limitation on how many overlays can be included in one briefing.

```
[OVERLAY0]
FILENAME=Slide0.cdtf

[OVERLAY1]
FILENAME=Slide1.cdtf

[OVERLAY2]
FILENAME=Slide2.cdtf

[OVERLAY3]
FILENAME=Slide3.cdtf
```

Figure 31: Example briefing summary file *.show.

Note that it is important that overlay headers are numbered beginning from zero and that there are no gaps between the overlay numbers. The briefing module will stop loading overlays if the next overlay number is missing.

APPENDIX B: THISTLE DEVELOPER GUIDE

A theoretical layout of the THISTLE software environment is given with the following graph.

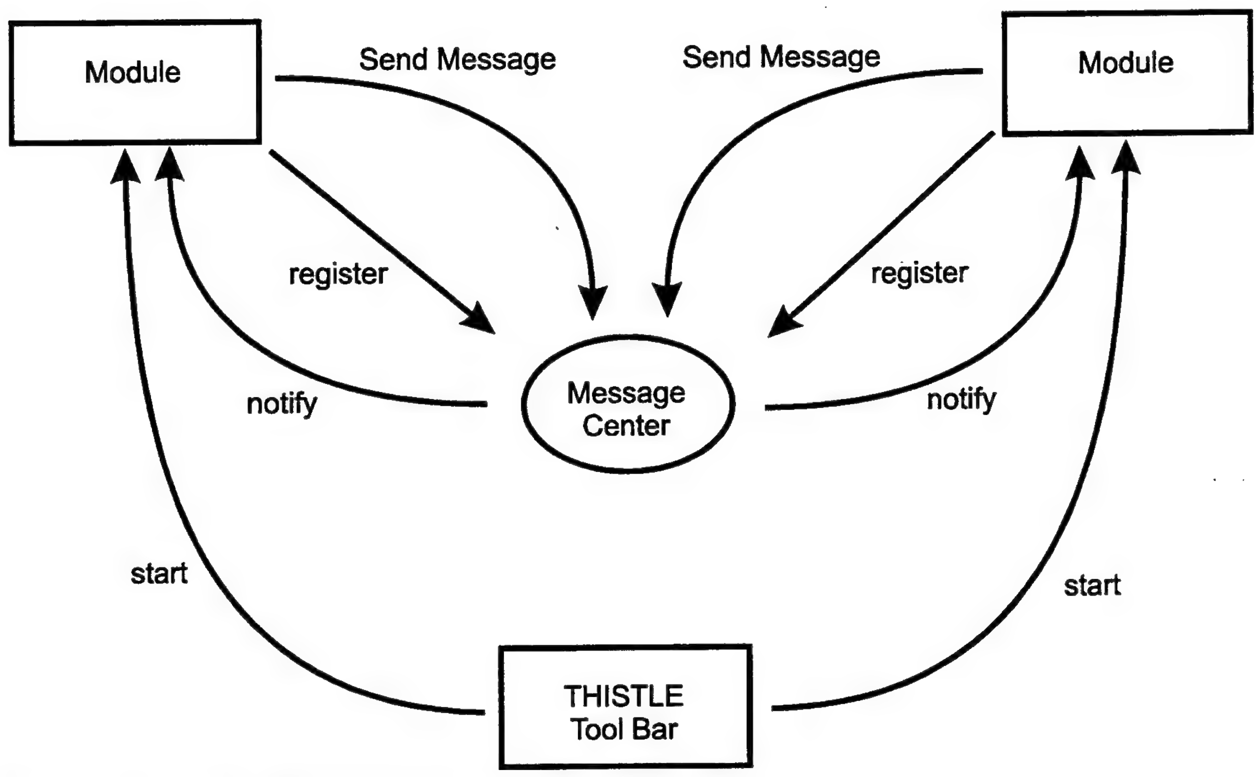


Figure 32: Theoretical THISTLE software architecture.

A. MESSAGE CENTER

The message center is the heart of the THISTLE software environment. It is implemented as a static class MessageCenter. This provides the advantage that it does not have to be instantiated by any of the software tools. The first time it is referred to by a THISTLE module it will exist automatically. The function of the THISTLE tool bar is to provide a easy way to start the individual modules inside the same Java virtual machine. It is absolutely essential that all modules are executed within the same virtual machine in order that the message center can operate correctly. All THISTLE modules can be complete stand-alone software products. They do not have to implement any particular interface in order to work within the THISTLE environment. However, they have to follow a certain set of rules so that they can communicate with the message center.

The THISTLE tool bar can only instatiate modules which provide a empty constructor. This means that the main method for these modules for stand-alone mode should be just an instantiation of itself. The same method is then used by the THISTLE tool bar to invoke these modules.

The sending of messages to the message center is handled by the following static method of the MessageCenter class.

```
MessageCenter.sendMessage(Object sendingObject, Object message);
```

This will send a message object to the message center. The sendingObject is a reference to the originating module of the message, which typically will be "this". Upon receipt of a message the message center will broadcast it to all its registered listeners.

A software module can register as a listener with the message center by calling the following static method of the MessageCenter class:

```
MessageCenter.addMessageListener (Object listener);
```

In general addMessageListener method should be called with "this" as the argument.

The message center will notify all registered tools every time it receives a message. However, it does not broadcast the message. The tools will be notified by calling the handleNewMessage(MessageEvent message) method. It is then up to the tool to retrieve the message and to handle it. Note that it is not required that a tool takes any action at all. If the handleNewMessage method of a registered listener cannot be invoked by MessageCenter, it will just be ignored. No warning or error message is thrown up in this case.

A message listener can remove itself from the list of listeners in the message center by calling the static method

```
MessageCenter.removeMessageListener (Object listener);
```

It will then not get notified of future messages.

B. SENDING OF OVERLAYS

For the display of a tactical situation on a map, NATO follows the rules given by APP-6, Military Symbols for Land Based Systems (SHAPE, 1986).

For Project THISTLE, the HQ ARRC developed a data transfer format, called CDTF. CDTF specifies a format for exchange of tactical information, which then can be displayed on a map. The format for this is defined in the paper about the Common Data Transfer Format (Schrepf, 1996).

There are three major classes which are used to implement overlays. The basic symbols are defined within the CDTFSymbol.class. The CDTF.class wraps the individual symbols up under one header and holds the additional meta information about an overlay, like originator, date, write permissions, etc..

For display purpose the CDTF is then wrapped up in the Overlay.class with also provides the method for displaying an overlay on the map display. The following example shows how to use these classes for generating an overlay and sending it through the message center to FLORA.

```
import thistle.message.*;
import thistle.flora.coord.*;
import thistle.flora.symbol.*;

public OverlayTest {

 public OverlayTest () {
     // generate the overlay wrapper.
     Overlay overlay = new Overlay();
     // generate the CDTF wrapper.
     CDTF cdtf = new CDTF();
     cdtf.setTitle("My Overlay");
     // generate the symbols in CDTF format.
     CDTFSymbol symbol = new CDTFSymbol();
     symbol.setSymbolHeader("21rd Mech Inf Bn| | |U|2|N|C|");
     // make mechanized infantry unit.
     // symbol.setFieldA("89 100 90 32 32 32 32");
     symbol.setFieldA(UnitSymbol.UNIT, UnitSymbol.INFANTRY,
                          UnitSymbol.ARMOURED);
     // make battalion size unit.
     symbol.setFieldB("37 32");
     symbol.setFieldT(" ");
     // give Symbol a actual location on the map.
     symbol.setActualLocation("FLORA|50000 60000|");
     // add the symbol to the CDTF wrapper.
     cdtf.addSymbol(symbol);
     // load the symbol into the Overlay wrapper.
     overlay.loadOverlay(cdtf);
     MessageCenter.sendMessage(this, overlay);
 }
 public static void main (String [] args) {
  new thistle.flora.FloraMain ();
  new OverlayTest();
 }
}
```

Figure 33: Example code for generating and displaying geographical related data on FLORA.

The symbol codes for defining a tactical symbol are given in the CDTF paper (Schrepf, 1996).

C. RECEIVING GRID LOCATIONS FROM FLORA

By selecting the edit mode from the tool bar, FLORA can be enabled to convert a mouse click on the map display into real world grid location. In this mode each mouse click is transformed in a FloraCoordinate and send to the Message center. From there each registered tool will be informed about the new message and can pick the grid location up if desired. An example code for picking up a grid location form FLORA is given below:

```
import thistle.message.*;
import thistle.flora.coord.*;
  .
  .
  MessageCenter.addListener(this);
  .
  .
public void handleNewMessage (MessageEvent evt) {
  // check if I am the source of this message.
  if (evt.getSource() != (Object) this) {
    // check which kind of message it is.
    if
(evt.getMessage().getClass().getName().equals("thistle.flora.coord.FloraCoordinate")) {
      FloraCoordinate coord = (FloraCoordinate) evt.getMessage();
      System.out.println ("FLORA : " + coord);
      System.out.println ("UTM : " + coord.getUTM());
      System.out.println ("Latitude/Longitude : " + coord.getGeo());
    }
  }
}
```

Figure 34: Example code for receiving and handling of messages.

It is important to note that all coordinates in FLORA are stored in 10 meter units relative to the project area. The methods getUTM and getGeo convert these internal coordinates into real world coordinate systems like UTM or Latitude/Longitude.

LIST OF REFERENCES

Ahuja, A., Thomas, L.M., Orlin, J.B., Network Flows, Theory, Algorithms, and Applications, Prentice Hall, Englewood, NJ, USA, 1993.

Arntzen, A., Software Components for Air Defence Planning, Master Thesis, Naval Postgraduate School Monterey, CA, USA, 1998.

Bradley, G.H., Buss, A.H., An Architecture for Dynamic Planning Systems Using Loosely Coupled Components, Proposal for Reimbursable Research, Naval Postgraduate School Monterey, CA, USA, 1997.

Brown, G.G., Vassiliou, A.L., Optimizing Disaster Relief: Real-Time Operational and Tactical Decision Support, Naval Research Logistics, Vol. 40, pp. 1-23, 1993.

Buss, A.H., Modeling with Event Graphs, Proceedings of the 1996 Winter Simulation Conference, D. Morrice, J. Charnes (eds), Coronado, CA, USA, 1996.

Buss, A.H., A Tutorial on Discrete-Event Modeling with Simulation Graphs, Proceedings of the 1995 Winter Simulation Conference, K. Kang, W. Lilegdon, D. Goldsman (eds), Arlington, VA, USA, 1995.

Chan, P., Lee, R., Kramer, D.,The Java Class Libraries, Second Edition, Volume 1, Addison-Wesley , Berkeley, CA, USA, 1998.

Chan, P., Lee, R., The Java Class Libraries, Second Edition, Volume 2, Addison-Wesley , Berkeley, CA, USA, 1997.

Flanagan, D., Java in a Nutshell, A Desktop Quick Reference, Second Edition, O'Reilly, Sebasopol, CA, USA, 1997.

Law A.M., Kelton W.D., Simulation Modeling & Analysis, second edition, McGraw-Hill Inc., New York, NY, USA, 1991.

Schrepf, N.J., Common Data Transfer Format - CDTF, Version 1.0, HQ ARRC, Mönchengladbach, Germany, 1996.

SHAPE, APP-6, Military Symbols for Land Based Systems, NATO unclassified, Belgium, 1986.

Weber, J.L., Using Java 1.2, Special Edition, QUE, Indianapolis, IN, USA, 1998.

Zukowski, J., Java, AWT reference, O'Reilly, Sebasopol, CA, USA, 1997.

ABREVIATIONS

ACE	Allied Central Europe
AFSOUTH	Allied Forces South
ARRC	ACE Rapid Reaction Corps
BH	Bosnia/Herzegovina
C2	command and control
C3	command, control and communications
CDTF	Common Data Transfer Format
HQ	Headquarters
IFOR	Implementation Force
JDK	Java Development Kit
Lat	Latitude
Lon	Longitude
MAG	Micro Processor Application Group
NATO	North Atlantic Treaty Organization
NC3	NATO Command Control and Communications
OAB	Operational Analysis Branch
OOTW	Operations Other Than War
SACEUR	Supreme Allied Commander Europe
SHAPE	Supreme Headquarters Allied Power Europe
UN	United Nations
UTM	Universal Transverse Mercator

SYNONYMS

FLORA	Geographical user interface for displaying of maps and tactical symbols
GIF	CompuServe Graphics Interchange Format
Ini file	Initialization file for software modules
IVY	Movement simulation
PNG	Portable Networks Graphic
THISTLE	Software architecture which allows loosely coupled components to interact with each other

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.. 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Cranfield University, Microprocessor Application Group 2
   Cranfield,
   Bedfordshire, MK 47 OAL, England.

4. HQ ARRC, Operational Analysis Branch 2
   BFPO 140,
   41179 Mönchengladbach, Germany.

5. Prof. A.H. Buss, Code OR/Bu... 3
   Naval Postgraduate School
   Operations Research Department
   Monterey, California 93943

6. Prof. G.H.Bradley, Code OR/Bz 1
   Naval Postgraduate School
   Operations Research Department
   Monterey, California 93943

7. Capt. Norbert Schrepf,... 3
   Kutzerstr. 133
   90765 Fürth
   Germany